Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

ВЫСШАЯ ШКОЛА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ

Направление: 09.04.04 Программная инженерия

Профиль: Робототехника

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

РАЗРАБОТКА И ИМПЛЕМЕНТАЦИЯ СИСТЕМЫ УПРАВЛЕНИЯ И СЦЕНАРИЕВ АВТОНОМНОГО ДВИЖЕНИЯ МОБИЛЬНОГО РОБОТА АВРОРА ЮНИОР

Студент 2 курса
группы 11-721
« »_____2019 г. Шабалина К. С.

Научный руководитель
PhD, профессор кафедры
интеллектуальной робототехники
« »_____2019 г. Магид Е. А.

Директор Высшей школы ИТИС
«___»_____2019 г. Хасьянов А.Ф.

Казань – 2019 г.

Содержание

ВВЕДЕНИЕ

В последние 20-25 лет появилась область робототехники, которая сосредоточена на разработке новых автомобилей, называемых интеллектуальными транспортными средствами (ТС). На сегодняшний день автомобиль является одним из самых важных изобретений двадцатого века. В мире насчитывается более 800 миллионов автомобилей, и ожидается, что это число удвоится в ближайшие десять лет [1]. Это обстоятельство повлекло за собой активную разработку методов автоматизации типовых задач, выполняемых людьми в процессе вождения.

Определим интеллектуальное транспортное средство как транспортное средство, оснащенное устройствами восприятия, принятия решения и контроля, которые позволяют автоматизировать задачи вождения. В задачи вождения включаются: безопасное следование полосам движения по правилам дорожного движения; объезд всевозможных препятствий, включая обгон других транспортных средств; отслеживание поведения окружающих объектов с оценкой вероятности возникновения опасной ситуаций на дороге; следование по маршруту, определённому пользователем. Такое решение разрабатывается для обеспечения безопасности и удобства пассажиров.

Интеграция интеллектуальных ТС в жизнь человечества находится на начальном уровне. Технологии и методы для беспилотных транспортных средств все еще находятся на стадии апробации в реальных условиях. Разработка интеллектуальных беспилотных автомобилей является популярным направлением исследований среди крупных автомобильных компаний BMW [2], Tesla [3], а также ИТ-компаний - Яндекс [4], Google [5], Baidu (проект Apollo) [6], Uber [7].

Применение автономных ТС в других задачах (помимо перевозки пассажиров) так же вызывает интерес крупных инвестиционных компаний. Исследования ведутся в таких отраслях экономики, как:

- сельское хозяйство,
- добыча полезных ископаемых,
- строительство,
- поисково-спасательные работы,
- многие другие [1] [8].

Основной мотивацией для автоматизации такого типа задачи как управление ТС является снижение расходов на персонал, увеличение количества времени работы и исключение негативных человеческих факторов. Вопрос о целесообразности применения труда человека или робота, не имеет универсального ответа и рассматривается индивидуально в каждом частном случае.

Важно отметить, что с течением времени технология роботизации ТС нашла применение в области менеджмента и логистики – системе управления цепями поставок (далее SCM система) [9]. В частности, в последнее время актуальным становится решение задачи последнего километра, т.е. доставка товара со склада непосредственно конечному получателю [10]. В связи с желанием сократить ресурсные и временные затраты на доставку в конечные точки традиционным способом (человек-курьер), транспортные компании рассматривают альтернативные подходы, использующие беспилотные транспортные средства: беспилотные летательные аппараты (БЛА) и беспилотные наземные роботы (БНР) [11].

В данной работе рассматривается решение задачи последнего километра на основе мобильного робота российской компании «Юниор», включающее создание реалистичной модели робота, и последующая апробация модели для решения задачи доставки. Этап создания модели включает в себя как компьютерное моделирование, так и разработку системы управления роботом, включая проектирование контроллеров приводов и системы автономной навигации и ориентирования. Подробное обоснование поставленных задач приводится главах 2 и 4.

Необходимо отметить, что для робота «Юниор» отсутствует компьютерная модель для моделирования реалистичного поведения. Производителем робота предоставляются только базовые пакеты для управления движением колес робота и доступа к показаниям бортовых датчиков робота для использования в системе ROS. Вследствие этого, данная работа несет существенную научно-практическую новизну, реализуя реалистичную модель робота, систему автономной навигации и пример технологического решения проблемы последнего километра.

Данная работа состоит из нескольких этапов:

1) *Обзор литературы*. Изучение и анализ научных работ по тематике беспилотного транспорта и методов решения интеллектуальных задач для мобильных роботов.
2) *Система управления цепями поставок.* Этот этап включает определение понятий и перечня решаемых задач. Подробный анализ задач, которые могут быть эффективно решены методами робототехники. Формулировка решения кейса доставки товаров на базе мобильного робота «Юниор».
3) *Модель мобильного робота «Юниор».* Этот этап содержит описание процесса моделирования робота для среды Gazebo, включая определение значения параметров ПИД регулятора для каждого из приводов.
4) *Система управления роботом «Юниор»*. Расширение модели робота набором нод, отвечающих за высокоуровневый контроль рулевыми и приводными колесами робота.
5) *Система автономной навигации мобильного робота «Юниор»*. Конфигурирование подходящих и реализация отсутствующих необходимых элементов системы автономной навигации для робота «Юниор».

6) *Моделирование решения кейса доставки*. Использование разработанной системы для моделирования варианта решения задачи последнего километра.
7) *Выводы и планируемые исследования*. Выводы о проделанной работе, направления последующих исследований и смежных тем.

1 ОБЗОР ЛИТЕРАТУРЫ

В настоящее время существует ряд исследовательских работ, посвященных автоматизации процессов SCM систем. Исследование [12] посвящено анализу действий компании Amazon, как пример успешной роботизации на производстве. Amazon – одна из крупнейших компаний, перестроивших свои склады в склад нового поколения, использующих автономных складских роботов. Издержки использования классического подхода в компании приводили к высоким затратам времени выполнения и отправки заказов клиентам.

Среди проблем, с которыми сталкивается компания-гигант, авторы выделяют следующие основные:

- большое время ожидания заказа;
- низкая эффективность и производительность систем поставок и оформления заказов;
- отставание от технологических инноваций компаний-конкурентов;
- и др.

Компания Amazon внедрила в складские отделы автономных мобильных роботов Kiva, которые значительно упростили сбор заказов, ускорили срок выполнения заказов, обеспечивая гибкость и масштабируемость роботизированной системы. Главным образом, быстродействие привело к общему повышению лояльности клиентов фирме Amazon и, как результат, повышению продаж. Авторы заключают, что подобная автоматизация может быть полезна и другим компаниям-гигантам в сфере электронной коммерции (англ. e-commerce).

Автор [11] проводит обзор существующих решений по автономной доставке товаров с помощью роботов. В работе представлены несколько типов

роботов: БНР, БЛА и их применение в конкретном кейсе[1]. Среди разработок автономной доставки с помощью беспилотных летательных аппаратов относят кейсы компаний Dominos, Alphabet (компания, владеющая Google), Amazon, UPS, DHL (почтовая компания). Автор отдельно выделяет БНР, предназначенных для доставки по пешеходной зоне – тротуару. Примерами таких проектов являются компания Starship Technologies [13], Marble [14], Kiwi [15]. Автор рассматривает множество актуальных проектов по автономной доставке товаров, которые развиваются в настоящее время. Таким образом, тематика последнего километра является востребованной и вошедшей на международный рынок по запросу экономики (англ. on-demand economy).

Примером реального случая разработки робота для решения проблемы последнего километра является работа [16]. Авторы предлагают решение проблемы в городских условиях с помощью БЛА (компании Intel, модель БЛА: Aero Ready to Fly), который действует по следующему принципу: БЛА находит первичное (приблизительное) место доставки с помощью GPS системы, конечное – с помощью метки, нанесенной на поверхность точки доставки. В качестве места посадки авторы выделяют балкон или крыльцо дома с меткой на стене / окне здания. Первоначальный концепт был протестирован в смоделированной среде ROS / Gazebo, далее успешно апробирован в полевых условиях. Авторы предоставляют исходный код разработки для других исследовательских групп в качестве основы для сторонних исследовательских проектов.

Авторы [17] представляют решение последнего километра с помощью робомобиля. В работе представлено описание компонентов системы: обнаружение пешеходов, локализация, навигация. В качестве транспортного средства используется машина для гольфа Yamaha G22E. В данной работе

[1] Под термином «кейс» понимается описание определенной проблемной бизнес-ситуации.

исследование сосредоточено на разработке автономной системы ТС для функционирования в условиях высокой городской загруженности. Авторы уделяют внимание процессу восприятия ТС используя нескольких датчиков (камера, лазерный дальномер, GPS, IMU) для точного определения присутствия других объектов на дороге в режиме реального времени. Анализ информации с датчиков способствует планированию оптимальных траекторий ТС избегая столкновений с другими участниками движения.

В свою очередь, исследование [18] сосредоточено не только на процессе разработки сервисного робота для больниц, но и на совершенствовании методов управления траекторией движения робота. Авторы отмечают, что современные мобильные робототехнические системы концентрируются только на точности положения робота, оставляя открытым вопрос контроля движения в опасных зонах (например, узкие пространства, лестницы). Для повышения уровня безопасного функционирования робота необходимо не только анализировать данные с датчиков самого робота, но и с датчиков, установленных в его окружении. В работе [18] описывается новая система – световая система коммуникации (англ. visible light communication system) для предотвращения ситуаций падения робота в опасных участках внутри помещений. Система коммуникации уведомляет робота о близости опасных мест, таким образом робот функционирует в замкнутом пространстве, выполняя задачи по доставке лекарств в палаты без риска падения. Авторы называют свою световую систему внутренним GPS и локализацией.

Многие проекты, предлагающие использовать роботизированные системы для решения задач автоматизации SCM систем, включают этап моделирования и проектирование индивидуального дизайна робота. Этот этап является ключевым начальным этапом разработки собственного решения; определение области применения робота и среда его функционирования напрямую определяют его внешний вид и тип ограничений его мобильности (т.е. передвижений) в пространстве. В рамках обзора литературы были рассмотрены несколько работ, посвященных дизайну платформ робототехнических систем и зависимость

конструкции робота от предполагаемой среды функционирования. Многие изученные работы авторов направлены на решение конкретных прикладных задач в сфере мобильной робототехники, в частности – конфигурации колес мобильной платформы, вопроса улучшения всенаправленных колес, разработке новых дизайнов робота. Вследствие этого некоторые авторы не регламентируют существующие проблемы робототехники в статье, но описывают своё решение о реализации мобильной платформы с заданными требованиями к платформе (требования к перемещению, методам построению карт и локализации).

В работе [19] авторы предлагают один из возможных вариантов конструкции - сборка недорогой, простой для понимания и небольшой по размерам платформы для применения в обучении и научных исследованиях, включает дизайн и реализацию роботизированной платформы согласно выработанным требованиям. Важность поставленной проблемы состоит в том, что у многих популярных мобильных платформ архитектура аппаратного уровня и набор интерфейсов сильно отличается, вследствие чего с каждым роботом студентам необходимо проходить медленный этап ознакомления с аппаратной и программной частями, прежде чем начинать какую-либо исследовательскую работу с данной моделью робота. На основе анализа существующих решений авторы разработали собственную платформу, которая содержит достаточное количество интерфейсов ввода-вывода для выполнения обучающих практических заданий. В работе раскрывается понятие мобильной платформы и разнообразие видов – в статье присутствуют примеры существующих платформ, включая ссылки на полное описание модели. Авторы описывают аппаратный и программный уровни разрабатываемой модели робота, сопровождая описание подробными схемами используемых компонентов и их краткого технического описания. Особо важным разделом является подробное описание набора датчиков и их интерфейсов, представленный в виде таблицы. Авторы схематично (преимущественно с помощью диаграмм и схематичных рисунков) продемонстрировали архитектурное решение, отобразив взаимодействие нижнего (аппаратного) и верхнего (программного) уровней платформы. Статья

опубликована в 2017 году и её подход является новым и интересным решением поставленной задачи, включая интеграцию робота в ROS (англ. Robot Operating System). Тогда как в большинстве случаев, интеграция робота в ROS отсутствует и становится еще одной задачей (в силу большого количества различий в конструкциях роботов).

Столкнувшиеся с проблемой большой сложности программного обеспечения для робота (вследствие наличия большого количества бортовых датчиков) авторы решают данную проблему при помощью интеграции системы ROS в свою разработанную платформу [20]. Достоинством данной статьи является раскрытие организации аппаратных компонентов платформы в виде схемы размещения, включая обозначение занимаемого пространства в роботе. Работа также содержит методику тестирования разработанной платформы: тест на синхронизацию поворота колес, движение по заданной траектории (точнее, по контуру квадрата). Представленная информация процесса сборки содержит подробные инструкции, с подробным обозначением использованных компонентов: модели аппаратных компонентов, версий пакетов ROS и ОС Linux. Единственным недостатком работы является недостаточная подробность в описании интеграции аппаратной шины в последовательный порт ОС в ROS.

Статья [21] содержит метод улучшения качества движения всенаправленной платформы подбором оптимальных параметров ПИД регуляторов, применяя алгоритм дифференциальной эволюции. Подход основан на использовании полученных решений для прямой и обратной кинематики рассматриваемой платформы для проведения численных экспериментов. Результаты моделирования анализируются для оценки качества текущего набора параметров ПИД регулятора. Несмотря на то что в этой работе не затронут вопрос о применимости метода для роботов произвольного типа, предлагаемое решение применимо для большинства роботизированных платформ. Следует отметить, что авторы, несмотря на получение оптимальных параметров исключительно из моделирования, утверждают, что подобный подход применим к любым системам и получит широкое распространение в инженерной практике.

Авторы [22] отмечают особый приоритет задачи навигации и локализации в неизвестной окружающей среде для роботов, выполняющих задачи в помещениях (например, жилое помещение, офисное пространство или больничный корпус). Они демонстрируют, что для эффективной навигации достаточно использовать RGB-D камеру и метод Visual SLAM (англ. визуальный метод одновременной локализации и картографирования). Данные с камеры RGB-D, помимо двумерного растра кадра, дополнены картой глубины (англ. depth map). Такая технология используется в камере Microsoft Kinect. Авторы приводят классификацию типа движения платформы в зависимости от вида используемых колес, приводят кинематическую модель разработанной всенаправленной платформы (использующую колеса Mecanum), частично описывают аппаратный уровень, приводят взаимодействие уровней системы управления (микроконтроллеры, интеграция с ROS, RGB-D камера). Авторы демонстрируют результат выполнения SLAM с помощью RGB-D камеры, приводят результаты экспериментов по точности полученных результатов. Такой метод построения карты и локализации общеизвестен, однако используется реже методов, использующих лазерные дальномеры.

Работа [23] посвящена разработке всенаправленной платформы на основе колес Mecanum для образовательных целей. Мотивация авторов заключается в определении целесообразности использования роботов в различных повседневных задачах (наблюдение, транспортировка объектов, перемещение объектов в рабочем кабинете), что требует от мобильного робота способности к быстрому маневрированию при перемещении и умения поиска эффективного пути к следующей точке плана выполнения задачи. Авторы решают эту проблему использованием всенаправленных колес Mecanum. Они приводят общее характеристики и особенности применения существующих типов колес (классических, всенаправленных и т.д.), приводят сравнительную таблицу с указанием плюсов и минусов для следующих типов колес: универсальные всенаправленные (англ. universal wheel), Mecanum (или шведское колесо), ведущие рулевые (англ. Powered steered wheel) и роликовые (англ. caster wheel).

Проведенный сравнительный анализ является полезной информацией для любого инженера, который заинтересован в проектировании роботизированной платформы для функционирования в заданных условиях. Помимо качественных характеристик для типов колес, авторы подробно описывают разработанную ими платформу: механические свойства колес, решения задач кинематики, состав и маркировка деталей (электроника, типы бортовых датчиков). Основная польза данной статьи заключается в подробном описании типов колес в наглядной таблице демонстрирующей достоинства и недостатки для каждого типа.

Продолжая тематику о типах колес, была рассмотрена работа [24]. Эта статья является обзорной и не содержит собственных разработок авторов, в отличие от предыдущих рассмотренных статей. Авторы отмечают особую популярность всенаправленных роботов среди всех мобильных и приводят все типовые структуры всенаправленных колес. К сожалению, в описании не указаны примеры задачи, для которой выбор данного типа колеса будет оптимальным. Тем не менее обзор содержит всю минимально необходимую информацию о каждом из типов всенаправленных колес, представлены схемы различных конфигураций колес в роботе и содержит дополнение сравнительной таблицы плюсов и минусов 4 типов колес (см. предыдущий абзац/предыдущую работу). Эта работа также содержит ценные описания и схемы организации управления платформой такого типа. Другое важное достоинство этой статьи заключается в наличии библиографии с множеством ссылок на конкретные практические применения каждого типа всенаправленного колеса, что служит начальной точкой обзора литературы для существующих платформ, потенциальных проблем присущих определенному виду платформ и способах их решения. Эта статья была применена в исследовательской работе составления базы знаний о типах колес и их конфигураций, что явилось вспомогательным материалом, использованным для написания научной статьи по теме представленной диссертации.

Авторы следующей статьи [25] описывают процессы анализа требований, проектировании и сборки всенаправленной мобильной платформы.

Содержательная практическая работа, содержит набор рекомендуемых практик для проектирования платформы в целях успешного участия в различных соревнованиях, как например, RoboCup (международные соревнования среди роботов) или соревнований тематики поисково-спасательной робототехники. Тип мобильной платформы, т.е. всенаправленная мобильная платформа, был выбран на основе анализа всех соревновательных роботов класса «мобильная платформа», как отмечено ранее, всенаправленная платформа предпочтительна любой другой неголономной. Авторы дают рекомендации к решению задачи кинематики робота и размещению аппаратных частей. Особенно полезным является примеры конфигурации колес и готовый чертеж панели робота для их монтажа. Также представлены использованные в процессе разработки эксперименты для проверки манёвренности платформы. Среди недостатков можно назвать отсутствие формальных объяснений как для конкретной конфигурации колес и на каких основаниях была выбрана конкретная форма панели для их монтажа. Качественным улучшением статьи была бы небольшая секция о методике выбора конфигурации колес, причинах непопулярности некоторых конфигураций.

Несмотря на всеобщую популярность всенаправленных колес, они имеют ряд недостатков, которые сложно компенсировать, в силу определенных механических свойств такого решения: они не могут функционировать в среде с грязной или неровной поверхностью, работа [24] отмечает дополнительный недостаток, как неэффективность при транспортировке тяжелых грузов. Для повышения эффективности и сохранения полезных свойств, авторы представляют свою модификацию всенаправленного колеса. Особенности такого колеса описывается в серии работ [26] [27], в которых раскрывается как принцип работы, так и методы размещения и управления. Принципиально новым является способность такого колеса на свободное перемещение по неровной поверхности с большей нагрузкой на колеса (англ. load capacity).

В качестве продолжения рассмотрения темы улучшения стандартов всенаправленных колес, авторы работы [28] предложили два способа улучшения

шведского всенаправленного колеса. Авторы данной статьи четко обозначают решаемую проблему: при передвижении по наклонной или неровной поверхности может возникнуть ситуация, в которой часть колес будут касаться поверхности ободом колеса, а не роликами (пассивные элементы шведского колеса), тем самым препятствуя дальнейшему движению платформы. Авторы подмечают, что изобретатель шведского колеса Илон предложил также несколько измененный механический дизайн для решения такой проблемы. Однако улучшение имеет побочный эффект - снижается эффективность колеса, как следствие использования передачи силы через вращающиеся ролики колеса [28]. Учитывая оба обозначенные недостатка вариантов колеса, авторы предлагают простой и сложный варианты решения. Сложный, вариант включает в себя изменение механики колеса включением в конструкцию специально вращающихся роликов. Достоинством данной работы является ее прикладная новизна и наличие примеров использования в роботах, однако авторы не указали на экспериментальное подтверждение работы улучшенной модели колеса; работа описывает только модель колеса, но без экспериментов невозможно подтвердить устранение изъяна классического шведского колеса. Отсутствует описание кинематики колеса, что является одним из ключевых недостатков статьи. Данная работа рассматривалась для исследовательской работы только как источник описания возможных проблем при применении колес Илона.

Идея об улучшении шведского колеса была также предложена в работе [29]. Авторы аналогично определяют главное ограничение в применении колеса, как возможность его использования только при движении по ровным поверхностям. Авторы предлагают два возможных решения; оба включают изменения механики колес. Достоинством данной статьи является наличие качественных и наглядных схем предлагаемых решений; однако авторы не претендуют на совершенство своих решений и описывают несколько недостатков. Данная работа рассматривается только как гипотетические решения - до тех пор, пока авторы не проведут экспериментального подтверждения.

Прикладные работы описывающие разработку мобильной всенаправленной платформы рассматриваются в статьях [30], [31]. Объединяет статьи использование шведского колес.

Авторы [30] проектировали аппаратную часть своей платформы для управления одним микроконтроллером, обеспечивающего полноценное всенаправленное движение с минимумом используемых компонентов. В аппаратной части авторы не используют никаких датчиков кроме энкодеров, интегрируемых в привод колес. В рамках своей будущей работы они планируют развивать платформу с целью создания дешевого модульного автономного мобильного робота для использования на объектах производства [30].

Авторы работы [31] также предлагают способы решить одну из главную проблем робототехники – высокой стоимости оборудования. Дороговизна роботов значительно ограничивает их использование в задачах, в которых риск его повреждения существенно высок. В статье приводится один из способов уменьшения стоимости – использования деталей роботов, напечатанных на 3D-принтере. Авторы предлагают модели для печати деталей мобильной всенаправленной платформы и руководство по сборке. Результатом является полностью напечатанный мобильный робот. Актуальность статьи обусловлена доступностью и дешевизной печати на 3D-принтере в настоящее время, что в условиях образовательных учреждений будет одним из самых оптимальных и экономически выгодных решений. Авторы демонстрируют, как такой способ позволяет быстро расширять область применения такого робота: пример создания и подключения простого манипулятора для работы над такими востребованными задачами, как открывание дверей или закручивание/откручивание креплений.

Авторы [32] решают на базе своей разрабатываемой платформы задачу организации системы удаленного управления. Система разрабатывалась с целью упрощения управления движением платформы с дополнительным графическим интерфейсом мониторинга и диагностики параметров работы [32]. Актуальность организации защищённых систем удаленного управления обусловлена еще и

необходимостью защиты от злоумышленников. Разработка автономных платформ в большинстве случаев включает в себя организацию удаленного доступа для отладки или управления вручную. Авторы рассмотрели несколько способов управления роботом и реализовали свою систему использующую ручной джойстик. Сильной сторона работы является формальное описание организации системы удаленного управления с помощью блок-схем.

2 СИСТЕМА УПРАВЛЕНИЯ ЦЕПЯМИ ПОСТАВОК

Управление цепями поставок (англ. supply chain management, SCM) - это способ организации контроля процесса производства и транспортировки товаров или услуг, начиная от поставок сырья и компонентов продукта до доставки продукта потребителю. Для контроля такой задачи организовывается граф организаций (т.н. «звеньев» в цепочке, англ. chain), участвующих в производстве или транспортировке готовой продукции, включая поставщиков сырья, фабрики, склады и точки продаж товара конечным потребителям.

Методы управления цепями поставок (SCM) в значительной степени основаны на методах и стратегиях промышленного проектирования, системного проектирования, управления операциями, логистики, закупок, информационных технологий и маркетинга, стремясь выработать комплексный подход. Фундаментальные вопросы, исследуемые в области управления цепочками поставок, основаны на методах теории управления и устойчивости для поиска решений для управления рисками [33].

Выделяют следующие ключевые процессы в цепочке поставок:

- Управление взаимоотношениями с клиентами
- Управление обслуживанием клиентов
- Управление спросом
- Выполнение заказа
- Управление производственным потоком
- Управление отношениями с поставщиками
- Разработка продукта и коммерциализация
- Управление возвратами.

Одной центральной задачей SCM систем является минимизация общих затрат, с учетом существующих конфликтов имеющимися между участниками цепочки. Примером таких конфликтов является взаимосвязь между отделом продаж, желающим иметь более высокий уровень количества запасов товаров

для удовлетворения потребностей клиентов, и складом, для которого более низкий уровень запасов необходим для снижения затрат ресурсов на хранение.

2.1 Задачи систем управления цепями поставок

Существует шесть компонентов традиционного управления цепями поставок.

1) Планирование – планирование и управление всеми ресурсами, необходимыми для удовлетворения спроса клиентов на продукт или услугу компании.
2) Поиск «источников» (англ. sourcing) – выбор поставщиков для предоставления товаров и услуг, необходимых для создания продукта. Установка процессов для мониторинга и управления отношениями с поставщиками. Ключевые процессы этого компонента включают заказ, получение, управление запасами и авторизацию платежей поставщиков.
3) Производство (англ. making) – организация мероприятий, необходимых для приема сырья, производства продукта, проверки качества, упаковки для отправки.
4) Доставка (или логистика) – координация заказов клиентов, планирование доставки, отправка грузов, выставление счетов клиентам и получение платежей.
5) Возврат – создание сети или процесса для возврата дефектных, избыточных или нежелательных продуктов.
6) «Включение» (англ. Enabling) – создание процессов поддержки для мониторинга информации по всей цепи поставок и обеспечения соответствия всем нормативным требованиям. Такие процессы включают в себя: финансы, человеческие ресурсы, ИТ, управление объектами, управление дизайном продукта, продажами и обеспечение качества.

2.2 Автоматизация задач SCM систем

На сегодняшний день мир приближается к переходу на новую парадигму Индустрия 4.0 (четвертая промышленная революция). Индустрия 4.0 характеризуется повсеместным внедрением киберфизических систем в жизнь человека, в процесс производства различных отраслей. К киберфизическим системам относят большие данные, робототехнику, дополненную реальность, симуляторы, облачные вычисления, интернет вещей (англ. internet of things, IoT), информационную безопасность [34].

Индустрия 4.0 относится к современной тенденции автоматизации и обмена данными в производственных технологиях. Некоторые задачи SCM систем могут быть автоматизированы следующим образом:

1. Автоматизированное производство на заводах;
2. Автономные грузовые машины, перемещающие товары на склады с обновлением их местоположения в реальном времени;
3. Автоматизированные склады, использующие мобильных роботов для обработки всех операций: от сбора до транспортировки продуктов по складу;
4. Отслеживание статуса заказа клиентом;
5. Автономные дроны в качестве доставщиков товаров в конечную точку [35].

Новые технологии управления цепями поставок включают робототехнику, интеллектуальный склад, большие данные, интернет вещей искусственный интеллект. Данные технологии в настоящее время влияют на каждый компонент цепи поставок, а именно: удовлетворение потребностей клиентов, поиск поставщиков, закупки, распределение, транспортировку и хранение на складах [35]. Прогресс способствует повышению эффективности, экономичности, сокращению времени выполнения заказа и экономии средств в цепи поставок [36].

Тем не менее, автоматизация задач поставок может привести к потере рабочих мест на низком (производственном) уровне. Таким образом, со временем персоналу необходимо повышать свою квалификацию и стремиться к компетенциям по использованию новейших технологий.

2.3 Кейс: проблема последнего километра

Термин последний километр (англ. last mile delivery) описывает задачу доставки товара из единой транспортной точки (англ. transportation hub) в конечную точку назначения. Последний километр является частью процесса планирования поставок и перевозок товаров со склада в их конечный пункт, являясь частью процесса SCM систем – сферы логистики и доставки. Востребованность доставки в конечный пункт только растет – все чаще совершаются покупки товаров онлайн, так как сфера интернет-продаж достигает максимума в своем разнообразии предоставления услуг (товары для дома, одежда, техника, продукты, лекарственные препараты и др.); клиенты ожидают доставку в кратчайшие или средние сроки. Таким образом, внутри этого процесса выделяют несколько главных проблем транспортировки товаров:

- большинство потребителей находятся вне дома, когда осуществляется доставка товара; вследствие этого товар может испортиться из-за погодных условий или быть украденным;

- дорогая транспортировка товара до конечной точки назначения при использовании услуг курьера на автомобиле / другом виде транспорта (большие затраты на ресурсы);

- организация оптимальной и эффективной доставки.

Однако больше всего влияния эта проблема имеет не на потребителей, а на бизнес, который занимается доставкой товаров и/или их продажей. Среди крупных фирм, столкнувшихся с задачей последнего километра выделяют компании-гиганты Amazon и Alibaba, FedEx, UPS, Walmart и др. В настоящее время есть несколько способов решения вышеперечисленных проблем.

1. Увеличение количества распределенных центров-складов в городе. Таким образом, сокращается расстояние между складом товаром и клиентами.

2. Взаимодействие с клиентом: своевременная и быстрая доставка товара, постоянное информирование клиента о статусе доставки товара («в стадии отправки», «в пути», «в процессе доставки»).

3. Возможность клиенту отслеживать в реальном времени статус своих заказов.

4. Оптимизация доставки. Главным образом включает в себя задачу усовершенствования планирования маршрута доставки товара для того, чтобы сократить общее время доставки, потребление топлива для транспортного средства. При этом алгоритм планирования должен учитывать: доступность курьера в данный момент, близость местоположения, весовую вместимость транспортного средства доставки и другие факторы.

5. Создание удобных точек с сейфами. Еще один способ обойти проблему дорогостоящих доставок единичных заказов – установка специальных шкафов-сейфов в доступных местах, как торговые центры, почтовые отделения, банки. Предоставляя специальный ключ или сканер штрих-кода, клиент сможет получить посылку безопасным и надежным способом.

6. Автономные транспортные средства. В настоящее время это один из способов расширения экономики «по запросу» (англ. on-demand economy, экономическая активность цифровых торговых площадок, технологических компаний, удовлетворяющих запросы потребителей немедленным предоставлением товаров и услуг). Хотя в настоящее время существуют нерешенные вопросы по использованию автономных машин в городских пространствах, вариант использования БЛА является одним из потенциальных способов решения проблем последнего километра.

Проанализировав существующие решения, хочется отметить, что пункты 1-3 и 5 имеют свою реализацию и пользуются активным спросом; в то же время, онлайн-отслеживание товара (пункт 3) в настоящее время уже является обыденностью и ключевым признаком адекватности транспортной службы. В

свою очередь, пункты 4 и 6 представляют собой концептуальные решения, не вышедшие повсеместно на мировой рынок; эти решения являются альтернативными способами решения проблемы, которые требуют развития определенных технологий и наличия глубоких компетенций в области робототехники, транспортной логистики, информационных технологий и программной инженерии.

Проблема последнего километра является актуальной темой для разработки качественных и новых решений, которые бы оптимизировали процесс доставки товаров и уменьшили затраты компаний на ресурсы доставки. Предлагаемым вариантом решения проблемы является внедрение в процесс доставки мобильного робота «Юниор» и программирование его на совершение автономной доставки от точки склада до конечной точки назначения.

2.4 Идея решения кейса

Данный раздел включает в себя: доказательство целесообразности концепта решения, описание предлагаемого решения, требования к решению, ограничения, накладываемые на реализацию решения.

2.4.1 Доказательство целесообразности решения

Одно из решений проблемы последнего километра является возможность использования беспилотных летательных аппаратов (БЛА). Такое решение имеет основное преимущество в том, что БЛА не зависят от дорожного трафика, особенностей наземных маршрутов доставки; это гибкое решение, которое не требует перемен в дорожной среде, не представляет критичной опасности для окружающих (при соблюдении техники безопасности со стороны человека). С другой стороны, БЛА сопровождено ограничениями в его применении. Во-первых, летательный аппарат жестко ограничен грузоподъемностью, что напрямую влияет на характер доставляемых товаров и вероятнее всего, сильно

сокращает ассортимент видов товаров для доставки. Во-вторых, БЛА в среднем может находиться в режиме полета 20-30 минут, что приблизительно соответствует одной доставке в конечную точку. В-третьих, исходя из ограничения по времени полета БЛА складывается серьезная проблема расхождения идеологии последнего километра с предложенным решением: доставка должна быть эффективной и охватывать несколько точек при низкой затрате ресурсов; БЛА же в данном случае при полном заряде может охватить одну – две смежные точки. В таком случае можно предположить, что на складе будет несколько десятков БЛА, распределенных по зонам доставки, однако главное ограничение остается прежним – склад должен находиться в некоторой близости к конечным точкам доставки, так как БЛА необходимо не только доставить товар, но и вернуться на склад. В-четвертых, остается открытым вопрос надежного крепления товара к БЛА, высоте полета БЛА в городских условиях, организации безопасности полетов нескольких БЛА в одной городской среде.

В случае использования наземного мобильного робота ситуация может быть качественно улучшена по всем параметрам. Мобильный робот также имеет ограничением в весовой нагрузке, но весовая нагрузка у БЛА существенно меньше. В мобильного робота может быть помещено сразу несколько товаров для доставки, что позволяет роботу осуществлять доставку сразу в несколько конечных точек, тем самым соответствуя идеологии последнего километра. Другое значительное преимущество – БНР обеспечен питанием от полноразмерного свинцово-кислотного аккумулятора, имеющего время работы 10 и более часов (в случае БНР «Юниор» используется свинцово-кислотный аккумулятор Delta DTM 1233 L с емкостью 33Ah, 10 часовым разрядом). За это время мобильный робот может совершить доставку в десятки точек, заменяя несколько БЛА с одиночными полетами и постоянной подзарядкой. Робомобиль «Юниор» является малогабаритным роботом, позволяющим ему проезжать различные узкие пространства, легко объезжать препятствия на пути, совершать парковку внутри небольшой площади.

Таким образом, использование наземного мобильного робота имеет экономическую и логистическую выгоду по сравнению с разработкой доставок с помощью БЛА.

2.4.2 Описание предлагаемого решения

Проект предполагает использование мобильного робота «Юниор» как беспилотное ТС для доставки товаров в несколько конечных точек назначения. На складе в робомобиль будут погружены товары для доставки, в системе робомобиля указаны точки доставки. Далее система обрабатывает точки доставок и планирует оптимальный маршрут для робота. Статус доставки товара будет обновляться в приложении у клиента, ожидающего заказ. Предполагается, что система мобильного робота будет обновлять статус заказа («ожидает доставки», «в пути», «ожидает клиента», «доставлено клиенту»), рассчитывать примерное время доставки, а также уведомлять о прибытии в точку доставки в приложении клиента. По завершении задания робомобиль возвращается на центральный склад.

Описание вида среды работы БНР. На текущий момент остается открытым вопрос выбора среды, в которой будет функционировать беспилотное ТС. Использование беспилотного ТС в городской среде до сих пор является опасным и правовое регулирование данного вопроса еще отсутствует. Но в качестве прототипа предлагается использование БНР на проезжей части с соблюдением правил ПДД. Причины такого решения следующие:

- БНР, функционирующий вне проезжей части (например, на тротуарах), является потенциальной угрозой для любых живых объектов, находящихся рядом с БНР; эта угроза намного выше, чем БНР, функционирующий на проезжей части.

- БНР, совершающий доставку в конкретную точку не по проезжей части, имеет риск увеличить в несколько раз время доставки из-за объезда

многочисленных препятствий на пути (в том числе пешеходов), из-за движения по неоптимизированному, избыточному пути.

- БНР, совершающий доставку в конкретную точку не по проезжей части, может потерять управление/ выйти из строя / застрять в поврежденных участках поверхности, по которой передвигается БНР (например, по тротуару, имеющему резкие перепады или поврежденную асфальтированную поверхность).

Также важно отметить, что БНР на проезжей части – это лучшее решение для обеспечения безопасности окружающих живых объектов (людей, животных), так как проезжая часть изолирована от их большей части. Изолирование является не единственным аргументом в пользу безопасности БНР – предлагаемая модель «Юниор» может развить лишь небольшую скорость движения – 10-15 км/ч. Хотя в будущем планируется развить эту скорость до 20 км/ч (чтобы «Юниор» не являлся помехой во время движения автомобилей по проезжей части), БНР не будет являться реальной угрозой для человека. Нельзя утверждать, что при такой небольшой скорости полностью исключается опасность такого БНР, однако при должной разработке и тестировании движения робота возможно минимизировать этот фактор опасности.

2.4.3 Требования к решению

При развитии идеи применения робомобиля «Юниор» были выдвинуты следующие требования к решению проблемы последнего километра.

- Система планирования маршрута должна создать маршрут движения БНР максимально оптимальным по времени и затрачиваемым ресурсам робота (заряда аккумулятора).

- Информационная система (ИС) по созданию заказов доставки должна информировать клиентов (пользователей) об изменении статуса их заказа через клиентское приложение. Статусы: «ожидает доставки», «в пути», «ожидает клиента», «доставлено клиенту». Дополнительно, ИС должна информировать клиента о приблизительном времени ожидания заказа.

- Робомобиль «Юниор» должен обладать соответствующим аккумулятором, позволяющим работать 10 – 15 часов без подзарядки.

- Робомобиль «Юниор» должен иметь в себе достаточно внутреннего пространства для погружения в него товаров (10-15 кг нагрузки).

- Робомобиль «Юниор» должен иметь автономный механизм открытия верхней крышки для получения доступа к товарам внутри робота.

- Робомобиль «Юниор» должен развивать скорость движения до 20 км/ч для исключения фактора помехи робота на проезжей части.

Ограничения, накладываемые на реализацию проекта. При имеющейся комплектации робомобиля «Юниор» робот не может вместить в себя большое количество товаров. Вследствие этого необходимо провести анализ расположения внутренних аппаратных компонентов робота, предположить их новое расположение и затем вычислить количество образовавшегося свободного места. Знание величин свободного пространства дает возможность решить задачу о рюкзаке для данного робота: «уложить как можно большее число ценных вещей в рюкзак при условии, что вместимость рюкзака ограничена».

Следующим ограничением является необходимость присутствия клиента в конечной точке при доставке товара. Робомобиль может подъехать к конечной точке, но выгрузить нужный товар сможет только клиент. Альтернативным решением, которое бы полностью освободило человека от участия в данном процессе – использование БЛА, который мог бы совершать короткие вылеты из отсека робомобиля с захваченным товаром и доставления его до дверей клиента (дверей дома / дверей подъезда / дверей ворот).

Последним ограничением является чувствительность робомобиля к поверхности дороги. Для робота необходима средняя ровная поверхность во избежание его повреждения и обеспечения беспрепятственного передвижения по поверхности.

В настоящее время в РФ есть определенные проекты по доставке товаров с помощью БЛА; доставки товаров с помощью мобильных автономных роботов нет, хотя концептуальное решение предлагает компания Аврора «Роботикс» в

виде грузового робота «Каргобот» [37]. Эта тематика является новой в РФ и не имеет повсеместной известности или популярности. Предложенный здесь концепт мог бы стать первым в РФ решением о роботизированной доставке товаров с помощью малогабаритных мобильных роботов.

2.4.4 Альтернативные решения

В настоящее время существует несколько стартапов, сосредоточенных на решении задачи последнего километра. Большинство этих проектов развиваются в США, Западной Европе, Китае.

Проект Marble представляет доставку товаров с помощью малоразмерного мобильного колесного робота, который передвигается по тротуару (см. Рисунок 2.4.4.1, а). Проект в настоящее время развивается в Сан-Франциско, США. В число инвесторов входят китайский гигант Tencent. Основанная в 2015 году, компания Marble работает над созданием полностью автономных роботов для доставки, которые могут ориентироваться в динамической среде городских тротуаров. Робомобиль использует камеры, лазерный дальномер и 3D карты города с высоким разрешением для навигации. В то время как компания впервые провела испытания своих роботов в сотрудничестве со службой доставки еды Eat24, роботы Marble имеют сменные грузовые отсеки для перевозки различных видов товаров: от продуктов питания до рецептурных лекарств.

Проект Nuro относится к категории крупногабаритных доставочных роботов в Кремниевой долине, США (см. Рисунок 2.4.4.2). Основатели проекта – бывшие инженеры-автогонщики компании Google. Робомобиль также оснащен необходимыми для навигации датчиками: лазерный дальномер, камера, радар. Таким образом, стоимость сборки оборудования будет значительно меньше, чем сборка обычного автомобиля с автоматическим управлением.

Проект Starship основан в 2014 году. Лондонская компания Starship Technologies (с инженерными подразделениями в Эстонии) является одной из известных компаний-поставщиков роботов для автоматизированной доставки

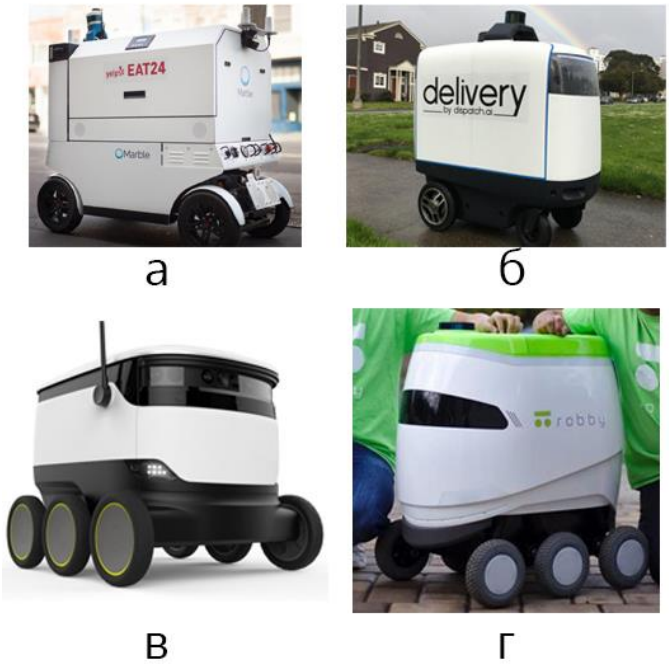

Рисунок 2.4.4.1 – роботы, используемые
для автономной доставки товаров

грузов. Компания Starship, основанная создателями Skype, создала своих малоразмерных роботов для автономной доставки. Клиенты заказывают товары через приложение, которое позволяет отслеживать состояние доставки роботом. Робот оснащен шестью колесами, имеют дальность действия две мили (~3,22 км), доставляют груз в течение 30 минут (см. Рисунок 2.4.4.1, в).

Проект Robby – другой проект Кремниевой долины, США (см. Рисунок 2.4.4.1, г). Пробег робота Robby 2 составляет более 20 миль (~32,2 км) без подзарядки, рабочая область робота составляет 8 городов Калифорнии, США. Робот Robby 2 укомплектован мощными инфракрасными камерами, которые способствуют безопасной навигации в темное время суток. Robby 2 спроектирован таким образом, чтобы он мог выполнять свои задачи независимо от погодных условий – его электроника находится под надежной защитой от влаги/воды.

Savioke, еще один стартап в Кремниевой долине, который использует роботов (см. Рисунок 2.4.4.2) для доставки кофе и других различных предметов внутри помещений: в отелях, производственных помещениях, медицинских учреждениях. Другие примеры роботов-доставщиков представлены на Рисунке 2.4.4.1, б.

Рисунок 2.4.4.2 – роботы Nuro (слева) и Savioke (справа) для автономной доставки товаров

3 ПРОГРАММНЫЙ ИНСТРУМЕНТАРИЙ

3.1 ROS

ROS (Robot Operating System) – популярный открытый фреймворк для командной разработки программного обеспечения роботов [38]. Атомарной единицей среды является пакет (англ. package), который должен решать определенную задачу. Согласно идеологии, ROS пакеты могут быть улучшены любым разработчиком, а новые пакеты – добавлены в общую базу знаний с описанием инструкции по использованию и составных модулей ROS.

Каждый пакет запускает свой процесс с индивидуальным наименованием. Процесс в ROS называется нодой / узлом (англ. node). Основная особенность ROS состоит в архитектуре в виде графов, где узлом является нода, создаваемая пакетом, а передача данных между узлами осуществляется через темы (англ. topic). Под передачей данных ROS подразумевает передачу сообщений (англ. messages) от одного процесса другому посредством тем. Каждый процесс может публиковать (англ. advertise) свои темы, отправляя на них сообщения с данными, а также подписываться на другие темы, «слушая» (англ. subscribe) сообщения от других процессов. В свою очередь, каждая тема поддерживает только один тип передаваемых сообщений.

ROS предоставляет следующие стандартные данные типов сообщений:

- Bool – логический тип данных.
- Int8, uint8, int16, uint16, int32, uint32, int64, uint64 – целочисленный тип данных с различной разрядностью.
- Float32, float64 – число с плавающей запятой с различной разрядностью.
- String – строковый тип данных.
- Time – тип данных время, выраженный через 32-разрядные числа без знака.
- Duration – тип данных промежутка, выраженный через 32-разрядные числа со знаком.

Дополнительно, ROS содержит свои собственные специализированные сообщения, которые необходимы для программирования роботов. Например, содержащийся в системе ROS пакет *turtlesim_node* реализует симуляцию робота в виде черепахи (черепаха – символ ROS системы). Этот пакет подписан на тему *turtleX/cmd_vel,* которая публикует сообщения типа *geometry_msgs/Twist* (одни из собственных сообщений ROS), описывающие линейную и угловую скорость черепахи TurtleX. Сообщение *geometry_msgs/Twist* состоит из следующих компонентов:

- *geometry_msgs/Vector3 linear*: float64 x, float64 y, float64z (описание линейной скорости)
- *geometry_msgs/Vector3 angular*: float64 x, float64 y, float64z (описание угловой скорости).

Другая возможность ROS – совместимость с различными симуляторами. Это позволяет разработчикам отлаживать свои ROS пакеты после проведения экспериментов в режиме симуляции.

3.2 Gazebo

Gazebo – открытый 3D симулятор, интегрированный с ROS. Gazebo предоставляет точную симуляцию различных типов существующих роботов (БНР, БЛА, манипуляторы), а также позволяет создавать собственные модели роботов. Симулятор учитывает заданную физику мира и модели, что позволяет проводить эксперименты (например, верификацию новых алгоритмов) с роботами в режиме симуляции. Также симулятор позволяет создавать желаемую среду для робота, добавлять различные препятствия и другие объекты. В дополнении, Gazebo предоставляет различные плагины, которые симулируют различные типы датчиков. Формат описания модели в Gazebo представлен в двух способах – SDF (Simulation Description Format) и URDF (Unified Robot Description

Format). Подробнее о URDF будет представлено в главе 4. Симулятор поддерживает STL и DAE форматы полигональных сеток (мешей) для объектов.

3.3 Rviz

Rviz – средство для 3D визуализации данных (сообщений), поступающих на темы (шины для передачи определенного типа данных) ROS. Визуализируя данные, разработчики могут оценить, как робот «видит» окружающую его среду и как он воспринимает себя в ней – его положение, ориентацию в пространстве, а также местоположение на карте. Rviz помогает в отладке программных компонентов ROS и является незаменимым инструментов при работе с роботом как в реальном окружении, так и в симуляции. Графический интерфейс ПО представлен на Рисунке 3.3.1.

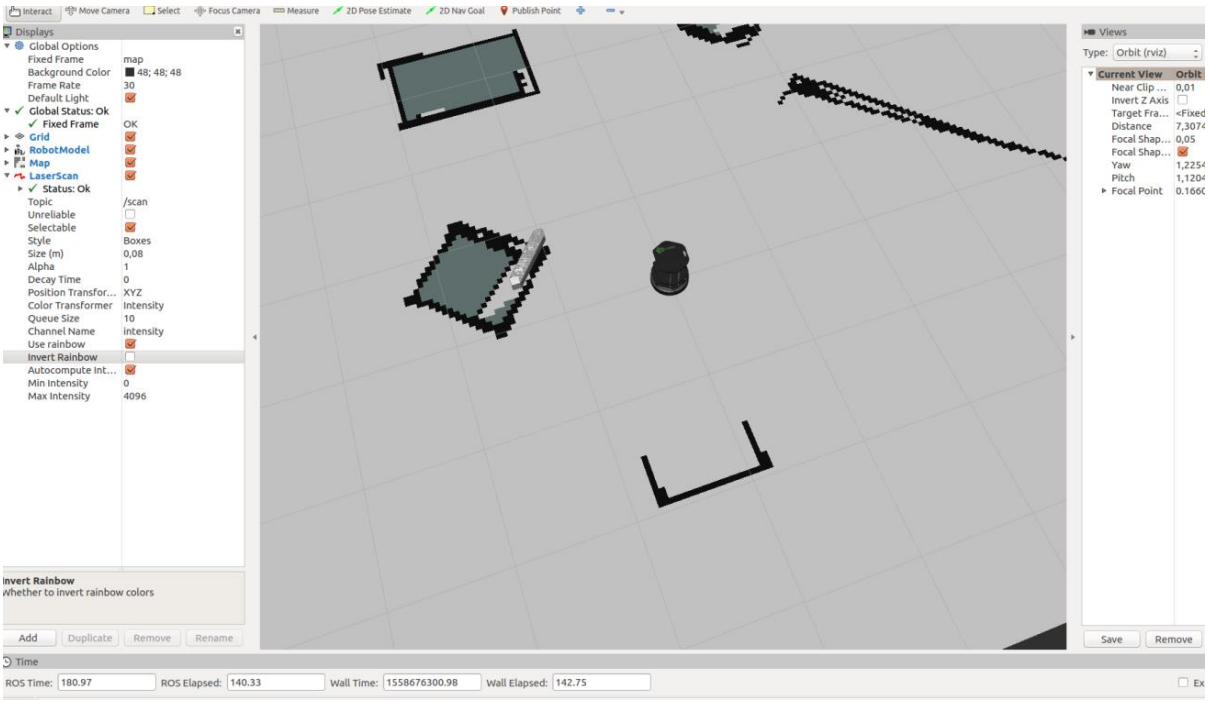

Рисунок 3.3.1 – Графический интерфейс Rviz

4 МОДЕЛИРОВАНИЕ МОБИЛЬНОГО РОБОТА «ЮНИОР»

В последние десятилетия широкое использование симуляций стало неотъемлемой частью исследований в области робототехники. Использование симуляторов помогает удешевить и ускорить разработку РТС (Робототехнических систем), является более безопасным инструментом первичной верификации идей и алгоритмов, а также позволяет тестировать новые концепции и алгоритмы, независимо от наличия требуемого оборудования у научной группы. Моделирование может применяться для всех типов РТС и задач на начальных этапах развития проекта.

Симуляторы стали прогрессивными инструментами, которые позволяют воспроизводить сложные среды и использовать определяемую пользователем физику, что дает возможность создавать модели РТС, которые ведут себя максимально близко к реальному поведению робота. Gazebo является одним из самых популярных 3D-симуляторов роботов, который успешно используется для симуляции БЛА [39] и БНР [40] [41] [42] [43], и для проведения различных экспериментов, включая тестирование базовых движений роботов, планирование пути и коллаборативное взаимодействие с другими роботами [44], эксперименты с манипуляторами [45] и моделирование сценариев поисково-спасательных операций [46]. Другие специализированные симуляторы (UWSim [47], V-REP [48]) позволяют моделировать водную среду для автономных подводных роботов.

В рамках задачи по созданию системы навигации, разработке алгоритмов передвижения (англ. locomotion algorithms) и автономной навигации для мобильного робота «Юниор» была построена модель в симуляторе Gazebo и проведено тестирование алгоритмов управления.

Моделирование робомобиля происходило в среде ROS Gazebo, подробное описание которой представлено в главе 3. Подход к моделированию был следующий: учет физических характеристик робомобиля, учет датчиков

робомобиля и их характеристик, использование CAD-моделей робомобиля для моделирования.

При создании модели использовались следующие среды: ROS, Gazebo. Использованные инструменты: САПР КОМПАС-3D, Solid Works. Формат описания модели: URDF.

В качестве САПР было использовано ПО КОМПАС-3D и ПО Solid Works. В задаче моделирования САПР был использован для экспорта необходимых мешей САПР-модели в среду Gazebo в формате STL. Также в САПР были задействованы инструменты измерения определенных деталей робомобиля для построения точной модели в симуляции. Для описания модели в Gazebo был использован формат URDF (Унифицированный Формат Описания Робота, англ. Unified Robot Description Format).

URDF представляет собой файл формата XML, который описывает робота с помощью определенных тегов. Существуют основные теги, которые необходимы для описания визуальной структуры робота (его «скелета»): link и joint, соединение (сочленение) и сустав, соответственно. Сочленения соединяются между собой с помощью суставов, представляя собой древовидную структуру модели робота. Другие теги предназначены для описания внутренних характеристик сочленений и суставов, добавления плагинов датчиков и их настройки, указания контроллеров суставов и их характеристик.

4.1 Мобильный робот Аврора «Юниор

Робомобиль «Юниор» является разработкой российской компании «Аврора Роботикс». Робомобиль изображен на Рисунке 4.1.1. По своему внешнему строению «Юниор» является классической машиной с системой приводов (англ. Ackermann drive) и рулевой рейкой для осуществления управления движением машины.

Данный робот создавался с целью обучения студентов и школьников научных секций основам разработки автономного движения беспилотных

наземных машин. Такой учебно-отладочный комплекс является удобным и оптимальным решением для работы студентов и школьников: малые габариты и вес машины (относительно полноразмерных машин, которых используют для разработки автономного функционирования компании Uber, Google, Яндекс), открытая программная система на базе ОС Linux Ubuntu 16.04 LTS и ROS (Robot Operating System), открытая аппаратная система. Открытая аппаратная система подразумевает свободный физический доступ к бортовым датчикам машины, устройству электрической цепи машины, возможности изменения аппаратного строя машины, его усовершенствования и замены любых компонентов (как датчиков, так и базовых элементов – колёс, аккумулятора, моторов колёс и т.д.).

С машиной предполагается использование специального полигона, построенного по типу реальной проезжей части с соответствующими элементами: дорожными знаками, светофорами, туннелем, дорожной разметкой и пешеходным переходом. Благодаря всему комплекту данного учебного комплекса студенты могут разрабатывать алгоритмы автономного движения машины в искусственной среде дорожного окружения.

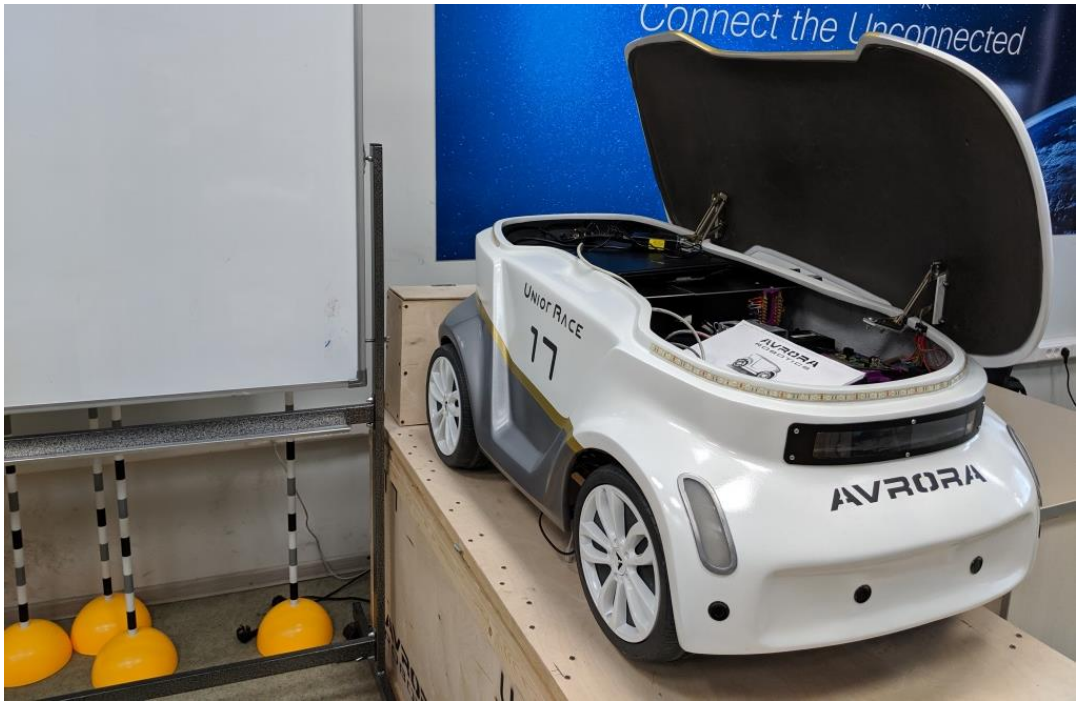

Рисунок 4.1.1 – Робот «Юниор»

4.1.1 Физические характеристики робомобиля

Физические характеристики представляют собой линейные размеры (см. Таблица 4.1.1.1) и весовые характеристики (см. Таблица 4.1.1.2) робомобиля «Юниор». Эти данные необходимы для создания корректной модели в симуляторе Gazebo. Точные весовые характеристики были запрошены у компании-производителя «Аврора Роботикс».

Таблица 4.1.1.1 – Линейные размеры робомобиля*

Наименование	Размер в сантиметрах	Размер в метрах
Диаметр диска колеса	20.5 см	0.205 м
Диаметр всего колеса	26 см	0.26 м
Продольное расстояние между колесами	71 см	0.71 м
Переднее и заднее расстояние между колесами	52.7 см	0.527 м
Продольная длина всей машины	110.5 см	110.5 м
Высота машины	53.5 см	0.535 м

*размеры указаны с точностью 0,1 см

Таблица 4.1.1.2 – Весовые характеристики робомобиля

Наименование	Вес (кг)
Колесо	0.9 кг
Внешний корпус робота	8.5 кг
Крышка робота	3.0 кг
Вес лазерного дальномера Hokuyo	0.16 кг
Вес камеры Kinect	0.45 кг
Вес всего робомобиля	43.5 кг

4.2 Моделирование основных элементов робомобиля

Основными элементами робомобиля, представляющими модель, были выбраны следующие: base_link (внешний каркас робота), right_steer_link (правая часть рулевой рейки, часть будущего механизма системы подвески робота согласно Ackermann drive), right_steer_wheel (переднее правое колесо), right_drive_wheel (заднее правое колесо), left_steer_link (левая часть рулевой рейки, часть будущего механизма системы подвески робота согласно Ackermann drive), left_steer_wheel (переднее левое колесо), left_drive_wheel (заднее левое колесо). Для того чтобы модель робомобиля визуально соответствовала

реальному роботу, были использованы меши частей робота для данных элементов и некоторых датчиков, рассмотренных ниже (САПР-модель была представлена компанией-производителем) (см. Рисунок 4.2.1).

Все сочленения модели были заданы в пропорциях, соответствующих пропорциям и размерам реального робота. Чтобы адаптировать инерционные характеристики робота к модели, был рассчитан упрощенный инерционный тензор для её сочленений. В процессе была заменена сложная структура меша каркаса робота (base_link) сплошным кубоидным объектом (англ. solid cuboid); структуры мешей рулевых колес (right_steer_wheel, left_steer_wheel), задних колес заменены на сплошной цилиндр (англ. solid cylinder) для вычисления приближенных тензоров инерции сочленений модели. Коллизии сочленений были указаны с учетом структуры мешей всех сочленений и в том же размере, что и сочленения. Файлы моделирования робота представлены в Приложении А.

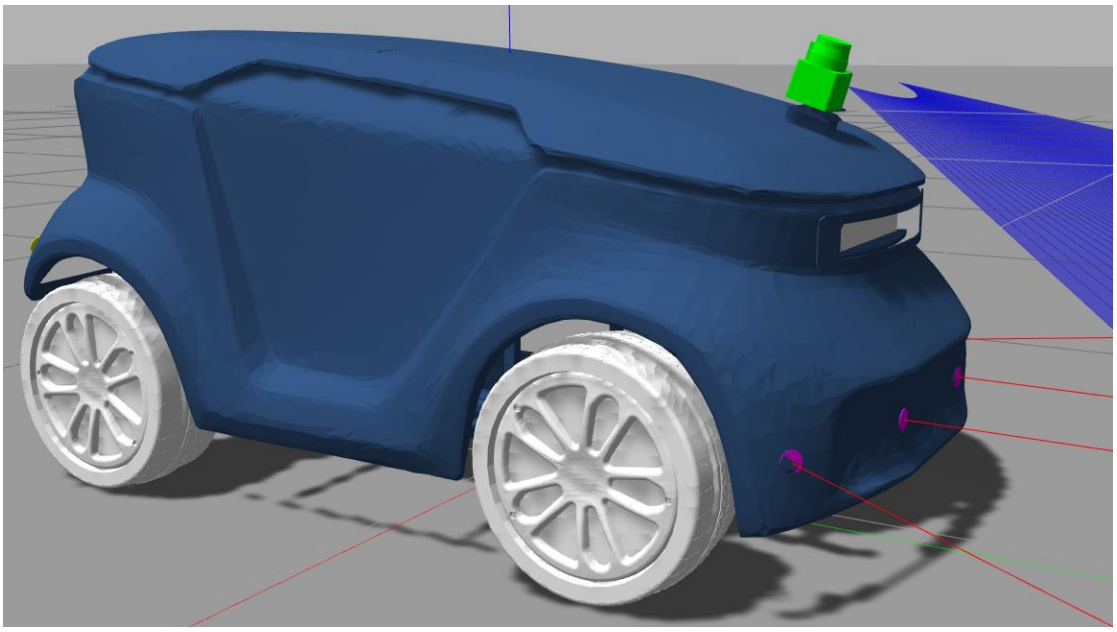

Рисунок 4.2.1 – модель робота «Юниор» в Gazebo

4.3 Суставы, моторы и трансмиссия модели

URDF формат позволяет описывать следующие типы суставов: призматический (англ. prismatic), вращательный с лимитами (англ. revolute), вращательный без лимитов (англ. continuous), «плавающий» (англ. floating) и

неподвижный (англ. fixed). Подробные определения суставов представлены ниже.

- Prismatic – скользящий сустав, который смещается вдоль заданной оси и имеет ограниченный диапазон движения, заданный верхним и нижним лимитами.
- Revolute – вращательный сустав (шарнирный), который вращается относительно заданной оси и имеет ограниченный диапазон движения, заданный верхним и нижним лимитами.
- Continuous – вращательный сустав (шарнирный), который вращается относительно заданной оси и не имеет верхних и нижних лимитов.
- Floating – сустав, имеющий движение во всех шести степенях свободы.
- Fixed – фиксированный сустав, который, в действительности, не является суставом. Все степени свободы такого сустава заблокированы, он не требует указания оси движения, лимитов. Такой сустав используется для жесткого сцепления сочленений робота между собой, а также присоединения датчиков робота к его сочленениям.

Вращательный тип сустава без ограничений был применен к суставам передних и задних колес (left_steer_wheel_joint, left_drive_wheel_joint, right_steer_wheel_joint, right_drive_wheel_joint). Вращательный типа сустава с ограничением был применен к рулевым механизмам модели (left_steer_joint and right_steer_joint), для последующей разработки системы подвески и управления моделью согласно принципу Акерманна.

Так как на реальном роботе используются два мотора для движения задних колес и один мотор для организации рулевого механизма, были сконфигурированы следующие ROS-контроллеры для суставов: контроллеры положения SteerRight_controller и SteerLeft_controller (ROS JointPositionController), контроллеры усилий EffortDriveRight_controller и EffortDriveLeft_controller (ROS JointEffortController). Контроллеры положения относятся к суставам рулевого механизма, контроллеры усилий относятся к суставам задних колес (так как на реальном роботе моторами оснащены два задних колеса). Рисунок 4.3.2 показывает финальную схему работы контроллеров в среде ROS.

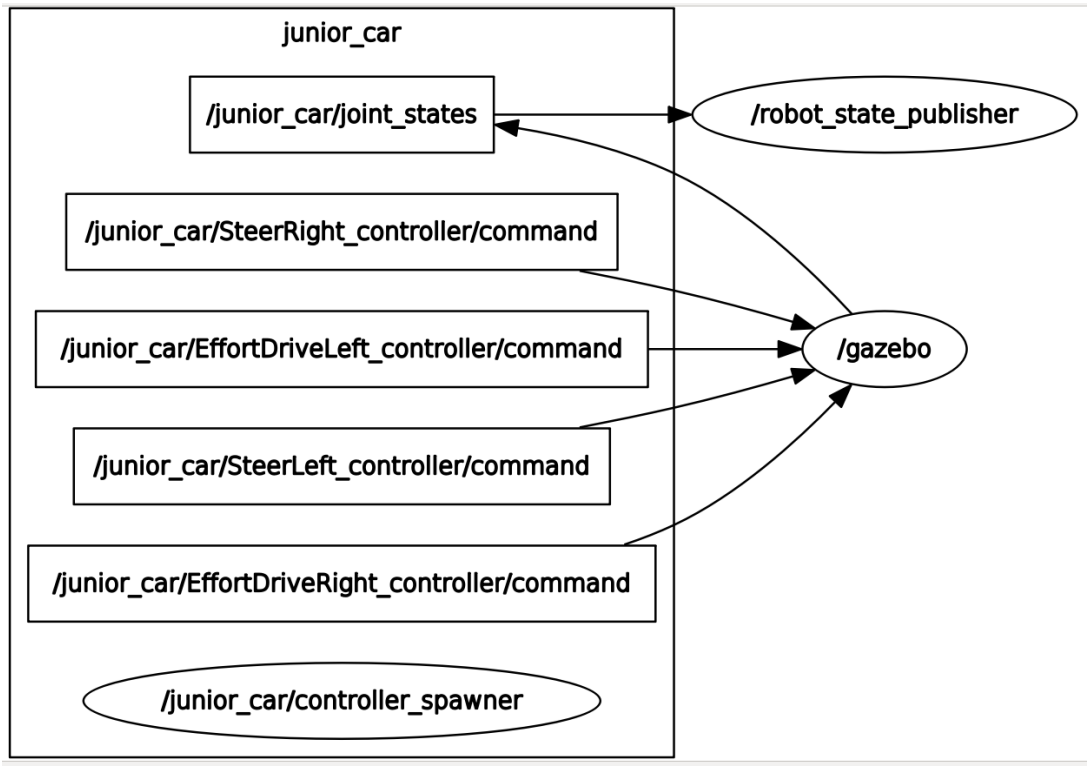

Рисунок 4.3.2 – Визуализация в rqt_graph тем сообщений модели робота «Юниор» (темы сообщений датчиков не визуализированы)

4.4 Симуляция датчиков

Таблица 4.4.1 представляет список датчиков робота «Юниор» и их имплементация в среде ROS и Gazebo. Модели датчиков были помещены в том же положении и ориентации, как и на реальном роботе, для получения точных

данных с датчиков. Все плагины датчиков были сконфигурированы согласно техническому описанию существующих датчиков. Для визуализации датчиков использовалось ПО Rviz (входит в фреймворк ROS).

Таблица 4.1.1 – Датчики робомобиля и их плагины в ROS, Gazebo средах

Датчик	Пакет в ROS	Плагин в Gazebo
Лидар	hokuyo_node	laser_controller
Microsoft Kinect	freenect_stack	camera_plugin
GPS	ur_nmea_driver	novatel_gps_sim
УЗ	ur_rangefinder_driver	sonar
Инерционное измерительное устройство	myahrs_driver	imu_plugin

4.5 Настройка ПИД регуляторов модели робота

Этап настройки ПИД регуляторов является продолжением работы по моделированию робомобиля. При описании контроллеров модели использовался конфигурационный файл, который указывал ROS и Gazebo тип используемого контроллера (в данном случае – position controller, effort controller) и параметры ПИД регулятора (пропорциональный p, интегральный i и дифференциальный d коэффициенты контроллера). В общих случаях в моделях роботов могут использоваться коэффициенты по умолчанию: p = 100.0, i = 0.01, d = 10.0; в случае же создания конкретной модели робота необходима индивидуальная настройка параметров контроллеров.

4.5.1 Понятие ПИД регулятора

ПИД контроллер (*регулятор)* – устройство, которое поддерживает определенные параметры объекта на заданном (желаемом) уровне. Регулятор следит за состоянием объекта и вырабатывает управляющие воздействия для того, чтобы обеспечить стабильность параметра регулирования. ПИД регулятор является пропорционально-интегрально-дифференцирующим регулятором. Он формирует выходной сигнал, являющийся суммой трех составляющих с

разными передаточными характеристиками. В формировании выходное сигнала ПИД регулятора участвуют:

- Пропорциональный коэффициент. Его значение пропорциональное ошибке *рассогласования* (разности заданного желаемого и реального значений регулируемого параметра).
- Интегрирующий коэффициент. Интеграл ошибки.
- Дифференцирующий коэффициент. Производная ошибки.

Работа ПИД регулятора может быть описана следующими шагами (см. Рисунок 4.5.1). Первый шаг – измерение реального значения регулируемого параметра. Второй шаг – получение ошибки рассогласования. Ошибка поступает на вход к ПИД составляющим регулятора. Третий шаг – в результате суммы составляющих образуется управляющее воздействие, которое подается на регулирующий элемент. Регулировка параметров системы может происходить с помощью одной (P, I регулятор), двух (PI, PD регулятор) или трех составляющих (PID регулятор).

P (пропорциональная) составляющая. В формировании выходного сигнала участвует пропорциональная составляющая. Выходной сигнал регулятора компенсирует отклонение регулируемого параметра. Выходной сигнал увеличивается, если ошибка рассогласования увеличивается; если ошибка равна нулю, тогда выходной сигнал регулятора также равен нулю. Другими словами, пропорциональная составляющая ориентируется на «настоящее», т.е. реагирует на текущее изменение в системе.

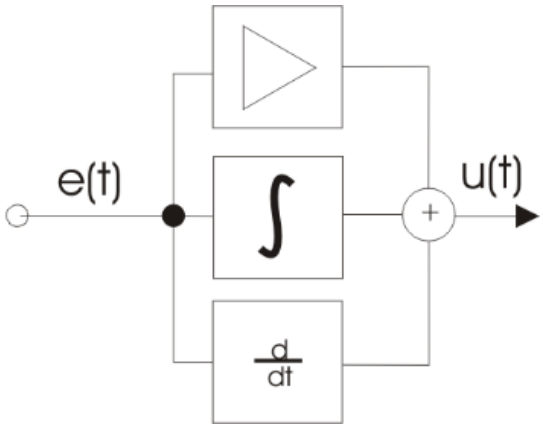

Рисунок 4.5.1 – Схема ПИД регулятора [61]

I (интегрирующая) составляющая. В выходном сигнале ошибка рассогласования умножается на коэффициент составляющей и прибавляется к предыдущему значению интегрирующего звена. Таким образом, выходной сигнал все время накапливается и со временем увеличивает свое воздействие на систему. Другими словами, интегрирующая составляющая ориентируется на «прошлое», т.е. постепенно влияет на систему, ориентируясь на прошлые состояния.

D (дифференцирующая) составляющая. Предсказывает отклонения регулируемого параметра в системе и в будущем противодействует этому отклонению.

Настройка ПИД регуляторов является сложной задачей. Во многих практических случаях коэффициенты ПИД определяется путем подбора, а также с помощью некоторых методик. Одна из методик, которая была использована для настройки ПИД регулятора модели, представлена ниже. Качество настройки ПИД можно определить по графику изменения регулируемого параметра во времени.

Общая концепция настройки ПИД регулятора. Составляющие регулятора настраиваются раздельно. Сначала отключаются (приравниваются к нулю) интегрирующий и дифференциальный коэффициенты. Настраивается пропорциональный коэффициент. Затем настраивается интегрирующий коэффициент, который должен минимизировать колебания параметра регулирования. Последним (при необходимости) настраивается дифференцирующий коэффициент. В большинстве случаев ПИД настраивается итерационно, т.е. до тех пор, пока не будет достигнут желаемый результат действия ПИД на регулируемый параметр системы.

4.5.2 Метод Зиглера-Никольса для настройки ПИД регулятора

Метод Зиглера-Никольса является эвристическим методом настройки ПИД-регулятора [49]. Последовательность настройки регулятора согласно методу, следующая:

1. Коэффициент интегральной и дифференциальной (I, D) составляющих приравнивается нулю.
2. Коэффициент пропорциональной составляющей Kp увеличивается с нуля до тех пор, пока не будет достигнуто значение предельного коэффициента усиления (англ. ultimate gain) Ku, при котором на выходе контура управления возникают стабильные и последовательные колебания.
3. Вычисляется значения периода Tu полученных стабильных и последовательных колебаний.
4. Ku и Tu используются для установки коэффициентов P, I, D в зависимости от типа используемого регулятора, согласно Таблице 4.5.2.1.

Таблица 4.5.2.1 – Метод Зиглера-Никольса

Тип регулятора	Kp	Ti	Td
П	0.5Ku	-	-
ПИ	0.45Ku	Tu/1.2	-
ПД	0.8Ku	-	Tu/8
Классический ПИД	0.6Ku	Tu/2	Tu/8
Интегральное правило Пессена	0.7Ku	Tu/2.5	3Tu/20
Допущение превышения желаемой границы	0.33Ku	Tu/2	Tu/3
Недопущение превышения желаемой границы	0.2Ku	Tu/2	Tu/3

4.5.3 Настройка ПИД регулятора для модели

Для настройки ПИД параметров модели был применен метод Зиглера-Никольса. Перед настройкой необходимо было запустить следующие модули ROS: *rqt_gui*, *rqt_recognifure*, запустить симуляцию модели в Gazebo и работу контроллеров. Интерфейс rqt_gui и подключенных плагинов изображен на Рисунке 4.5.3.1. Интерфейс *rqt_recognifure* изображен на Рисунке 4.5.3.2. Пакет *rqt_gui* предоставляет удобный способ (GUI интерфейс) для отправки сообщений (*команд*) на контроллеры модели, а также с помощью подключенного плагина Mat_plot отслеживать состояние контроллера при отправке сообщений. Модуль *rqt_recognifure* предоставляет пользовательский интерфейс для настройки ПИД параметров всех контроллеров модели.

Согласно методу Зиглера-Никольса, составляющие I и D были приравнены к нулю, а составляющая P регулятора постепенно увеличивалась. По достижении предельного коэффициента усиления Ku, выраженного на графике в виде стабильных колебаний (см. Рисунок 4.5.3.3). После вычисления периода колебаний, полученные данные использовались для расчета коэффициентов Kp, Ti, Td, согласно Таблице 4.5.2.1 (метод Зиглера-Никольса).

В итоге были получены следующие коэффициенты для контроллеров положения рулевого механизма (*SteerRight_controller* и *SteerLeft_controller):*

- Ku = 300.0
- Tu = 3 секунды
- Kp (пропорциональная составляющая) = 180.0
- Ti (интегральная составляющая) = 1.5

- Di (дифференциальная составляющая) = 0.375.

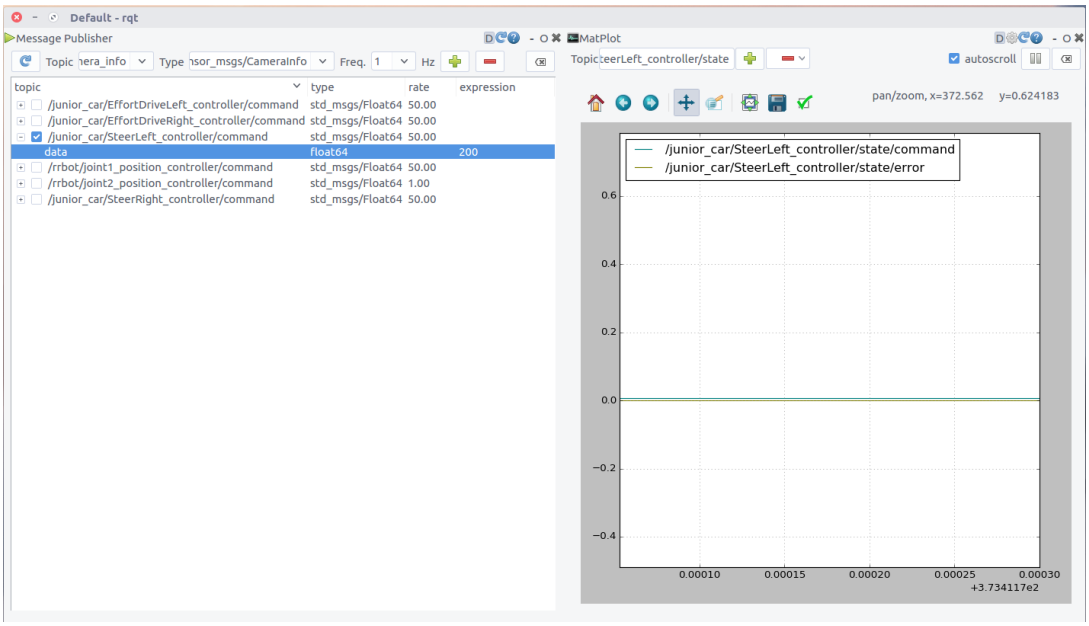

Рисунок 4.5.3.1 – Модуль rqt_gui в ROS

Следующим этапом работы потребовалась дополнительная настройка ПИД регуляторов для получения точных значений коэффициентов ПИД. Для этого аналогично был использован rqt_gui, rqt_recognifure, модель, запущенная в Gazebo. При дополнительной настройке с помощью rqt_recognifure поочередно были настроены все коэффициенты регулятора. Точность настройки определялась наличием стабильного поведения системы и временем затухания колебаний. Внешне это проверялось поведением модели робота в симуляторе Gazebo: при заданном на контроллер значении, сустав должен был прийти в указанное ему положение за допустимое время (время затухания колебаний). Так как при отправке сообщений на рулевую рейку реального робота, его передние колеса поворачиваются за 1 секунду, это время было принято, как допустимое время затухания колебаний для модели робота. Окончательные значения ПИД регулятора для передних колес, удовлетворяющие поставленным условиям, следующие:

- Kp (пропорциональная составляющая) = 380.0
- Ti (интегральная составляющая) = 50.5

- Di (дифференциальная составляющая) = 14.375.

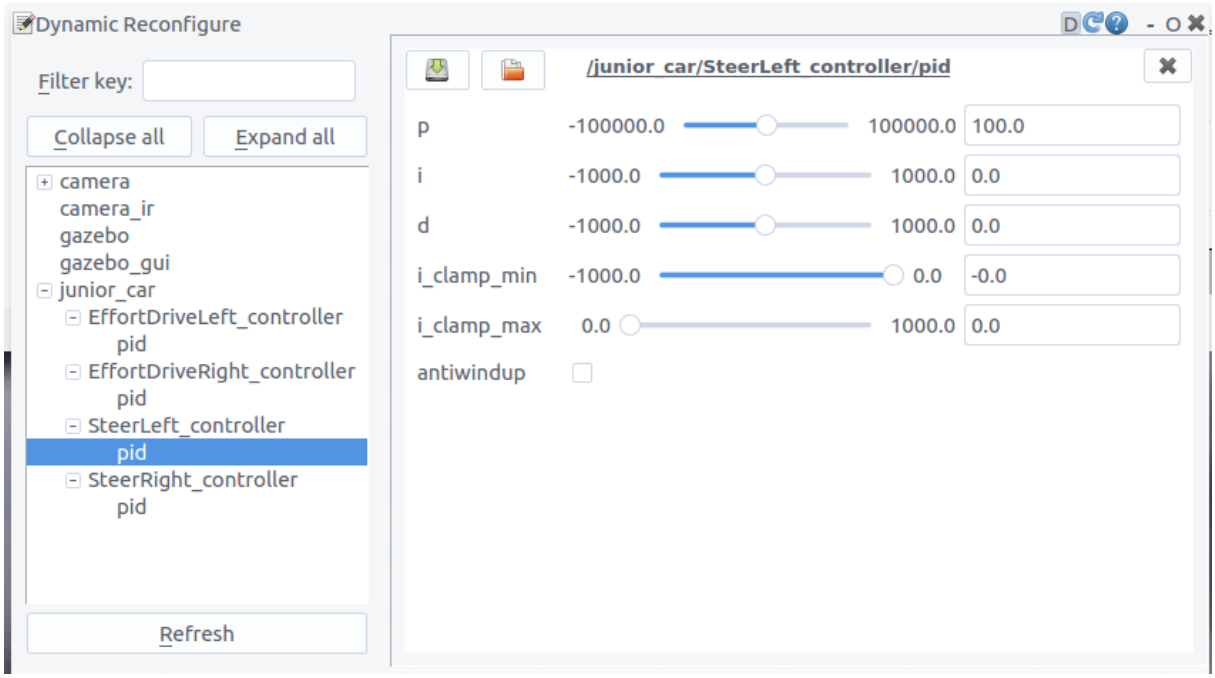

Рисунок 4.5.3.2 – Модуль *rqt_reconfigure* в ROS

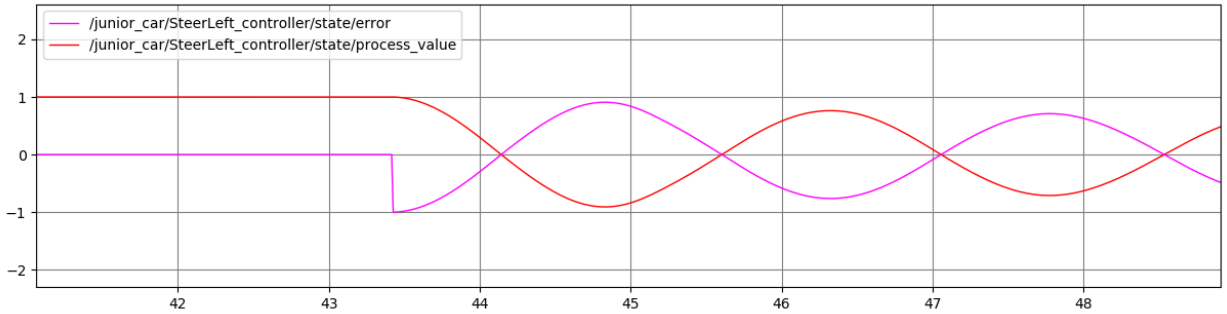

Рисунок 4.5.3.3 – Процесс настройки ПИД регулятора. Достижение Ku параметра (стабильных колебаний). Розовым цветом визуализирована текущая ошибка, красным – непосредственное значение величины сустава (его позиция) в текущий момент времени

5 ОРГАНИЗАЦИЯ КОНТРОЛЯ МОДЕЛИ РОБОТА «ЮНИОР»

В данной главе описан принцип рулевого управления и геометрии Акерманна и его реализация в ROS для виртуальной модели робота «Юниор».

5.1 Принцип рулевого управления и геометрии Акерманна

Мобильный робот «Юниор» имеет систему рулевого управления, аналогичную системе полноразмерных машин с задним приводом. У «Юниора» приводными колесами являются задние колеса, а передние – отвечают за направление движения робота. В случае полноразмерных автомобилей, рулевое управление осуществляется по следующей общей схеме (см. Рисунок 5.1.1).

С помощью рулевого колеса (отмечено номером 6 на Рисунке 5.1.1) задается желаемое направление движения автомобиля. В свою очередь, направление поворота рулевого колеса передается через рулевой вал на рулевую рейку (отмечено номером 3 на Рисунке 5.1.1). Движение рулевой рейки способствует движению рулевой поперечной тяги (отмечено номером 1 на Рисунке 5.1.1), на наконечниках которой находятся поворотные кулаки. В поворотные кулаки вставляются ступицы колес, тормозные диски и колеса автомобиля. Поворотные кулаки, в зависимости от движения рулевой рейки совершают поворот, тем самым направляя движение передних колес.

У робомобиля «Юниор» роли рулевого колеса и рулевой рейки объединены в один рулевой мотор. В зависимости от направления вращения рулевого мотора изменяется направление рулевых колес робота. Поворот рулевых колес происходит производится согласно геометрии Аккермана, которая соблюдается во многих современных автомобилях.

Геометрия Аккермана достигается механическим способом, с помощью правильного построения рулевой трапеции. С помощью рулевой трапеции передние колеса при повороте поворачиваются на разный угол; как правило,

внешнее колесо поворачивается на меньший угол, чем внутреннее колесо. Это происходит для того, чтобы уменьшить износ внутренней шины колеса и избежать потерю сцепления с дорогой при повороте.

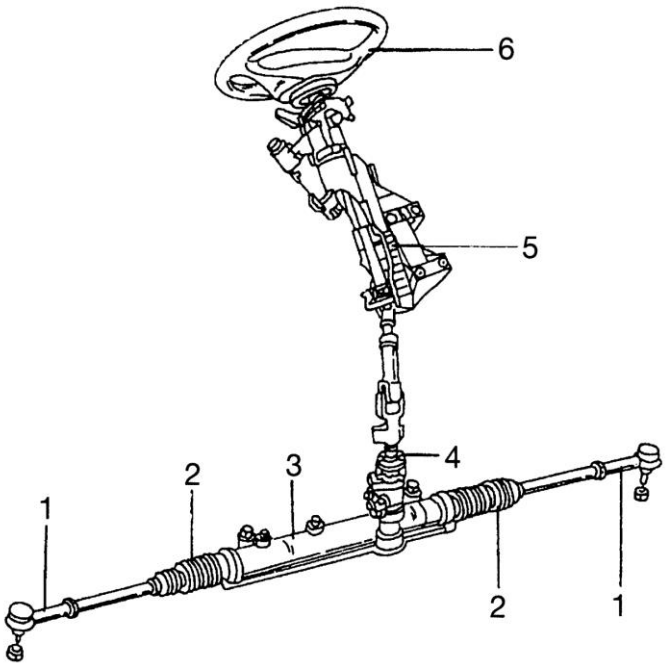

Рисунок 5.1.1 – Система рулевого управления автомобилем [60]. 1 – рулевые поперечные тяги, 2 – манжеты, 3 – рулевой механизм (рулевая рейка), 4 – эластичная муфта, 5 – рулевая колонка, 6 – рулевое колесо

5.2 Разработка контроля рулевого управления модели робота

Для реализации задачи был задействован классический (общий) случай расчёта геометрии Акерманна. Таким образом, для совершения вычислений угла поворота передних колес (α1 и α0), необходимо было знать линейные параметры мобильного робота: расстояние между передними колесами (T, T = 0.54 м) и боковое расстояние между передним и задним колесом (L, L=0.7 м). Эти параметры были использованы как константы при расчете углов поворота колес.

Согласно принципу геометрии Аккермана, общий угол α поворота всего мобильного робота может быть определен следующим соотношением (1):

$$\tan(\alpha) = \frac{L}{R} \qquad (1)$$

где R – ортогональное расстояние от продольной оси мобильного робота до центра поворота (см. Рисунок 5.2.1).

Углы поворота передних колес α1 и α0 определяются соотношениями (2) и (3):

$$\alpha_1 = tan^{-1}\left(\frac{L}{R-\frac{T}{2}}\right) \qquad (2)$$

$$\alpha_1 = tan^{-1}\left(\frac{L}{R+\frac{T}{2}}\right) \qquad (3)$$

В свою очередь, угол поворота α мобильного робота определяется заранее как входное значение для вычислений углов Аккермана; таким образом, радиус R поворота мобильного робота из отношения (1) может быть рассчитан заранее (4):

$$R = \frac{L}{\tan(\alpha)} \qquad (4)$$

Используя эти соотношения, была изменена нода steer_remap (контроль передних рулевых колес модели). Измененный принцип работы управлением рулевых колес модели стал следующим: на вход ноде поступает желаемый угол поворота всей модели (α, выражение 1). Угол задается в радианах, в зависимости от знака («+» или «-») определялось направление поворота вправо или влево, соответственно. В зависимости от желаемого поворота всей модели просчитывались индивидуальные углы поворота рулевых колес согласно геометрии Аккермана (2) и (3); внешнее рулевое колесо поворачивается на меньший угол, чем внутреннее рулевое колесо при совершении поворота модели. При расчете учитывались линейные размеры робота «Юниор». Исходный код разработанной ноды представлен в Приложении Б.

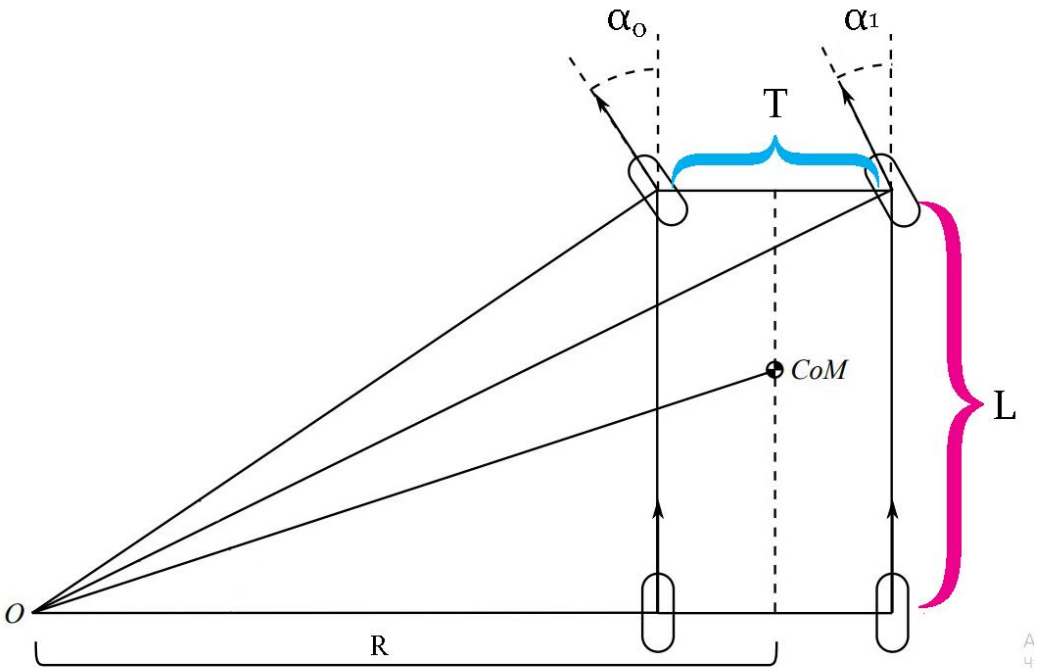

Рисунок 5.2.1 – Геометрия Акерманна для транспортного средства. R – расстояние от продольной оси ТС до центра поворота, L – боковое расстояние между серединой переднего и серединой заднего колес, T – расстояние между серединой передних колес

6 СИСТЕМА НАВИГАЦИИ ДЛЯ МОБИЛЬНОГО РОБОТА «ЮНИОР»

Данная глава описывает процесс создания и настройки системы навигации для модели робота «Юниор». Система навигации состоит из общих ключевых компонентов, описанных далее. В свою очередь, каждый компонент требует индивидуальной конфигурации, которая зависит от типа робота. Для создания системы навигации модели робота «Юниор», потребовалось следующее:

• реализация всех обязательных компонентов системы навигации, перечисленных далее,

• изменение ноды контроля рулевых колес согласно геометрии Акерманна,

• создание связи между cmd_vel системы навигации и текущих нод управления моделью,

• корректировка данных с лазера, установленного на модель робота,

• настройка параметров move_base.

6.1 Система навигации в ROS

Система навигации - это комплекс взаимосвязанных модулей, которые позволяют роботу строить карту окружающей среды, ориентироваться в построенной карте и просчитывать путь к поставленной на ней целевой точке (англ. goal point). Система навигации получает информацию из одометрии, датчиков и выдает команды скорости для отправки их на мобильную базу робота. Для правильной работы системы навигации ROS робот должен быть настроен следующим образом (требования к аппаратному обеспечению робота):

1. Робот должен быть дифференциально-колесным или голономным. Предполагается, что мобильная база робота управляется путем отправки желаемых команд скорости в виде скорость – x, скорость –y, скорость – θ.

2. Лазер должен быть установлен на плоской ровной поверхности, параллельной земле. Лазер используется для построения карты и локализации.

3. Система навигации ROS разработана для робота квадратной формы, поэтому его производительность будет наилучшей для роботов, близких к квадратной или круглой форме. Система навигации работает с роботами произвольной формы и размеров, но могут возникнуть проблемы для роботов прямоугольной формы при навигации в узких пространствах (например, в дверных проемах).

Кроме аппаратных требований, Система навигации состоит из комплекса взаимосвязанных программных пакетов ROS. Рисунок 6.1.1 отображает стандартизированную схему системы навигации для любого робота. Каждый элемент (кроме amcl и map_server) является обязательным и незаменимым в стеке навигации – при отсутствии какой-либо части Система навигации не будет функционировать. Компоненты в диаграмме, оформленные в белом цвете, являются реализованными и готовыми к использованию, в сером цвете –

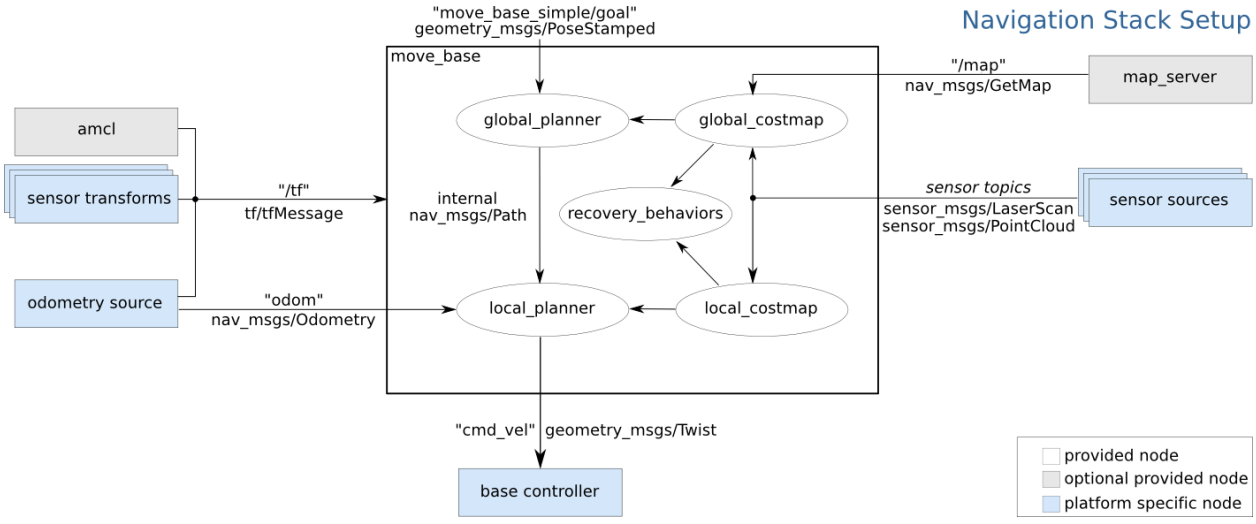

Рисунок 6.1.1 – Стек навигации в ROS [59]

реализованные и опциональные в использовании, в голубом цвете компоненты должны быть разработаны для каждой робототехнической платформы.

6.1.1 Модуль move_base

Модуль move_base является основной частью системы навигации робота. Он получает данные с остальных компонентов системы и использует их для анализа и принятия решений в процессе навигации робота. Move_base координирует поступающие данные и посылает соответствующие команды на контроллер базы (через cmd_vel), приводя в движение робота. Запуск и конфигурация move_base для модели робота «Юниор» описаны в Приложении Г.1, Г.2.

Этот модуль включает в себя следующие подмодули:

- Global_planner, глобальный планировщик.
- Local_planner, локальный планировщик.
- Local_costmap, локальная карта стоимости. Карта, которая хранит информацию о препятствиях в мире робота. Данная карта используется для локального планирования и обхода препятствий.
- Global_costmap, глобальная карта стоимости. Карта, которая хранит информацию о препятствиях в мире робота. Глобальная карта используется для создания долгосрочных планов по всему окружающему пространству робота.
- Recovery_behaviros, поведение восстановления. Этот подмодуль сочетает в себе несколько мероприятий, проводимых в случае потери плана (пути) у move_base. Обычно, этот подмодуль приводится в действие, когда робот воспринимает себя, как застрявшего на карте.

Все модули move_base настраиваются в специальных конфигурационных файлах формата «*. yaml», а move_base запускается с помощью launch-файла.

6.1.2 Необходимые компоненты системы навигации

Конфигурация преобразований. Система навигации требует, чтобы робот публиковал информацию об отношениях между координатными системами внутри робота, используя tf ros-пакет.

Tf (Transformation) – это пакет, который позволяет пользователю прослеживать несколько координатных систем с течением времени. Tf отражает отношения между координатными системами в виде древовидной структуры (буферизированной по времени) что позволяет пользователю вычислять координаты вектора в любой координатной системе для любого момента времени.

Данные с датчиков. Система навигации использует информацию с бортовых датчиков робота для избегания столкновений с препятствиями; Система навигации предполагает, что бортовые датчики публикуют либо sensor_msgs/LaserScan сообщения (что означает сообщения с лазерного дальномера), либо sensor_msgs/PointCloud (что означает сообщения с камеры глубины) в ROS.

Информация об одометрии. Система навигации требует публикации информации об одометрии робота с использованием tf и сообщений nav_msgs/Odometry.

Контроллер базы (англ. Base controller). Система навигации предполагает, что он может отправлять команды скорости с помощью сообщений geometry_msgs/Twist; в свою очередь, эти сообщения должны находиться в рамках темы cmd_vel. Это означает, что должна существовать нода, которая слушает тему cmd_vel, принимает команды типа vx, vy, vtheta (cmd_vel.linear.x – линейная скорость в оси x, cmd_vel.linear.y – линейная скопрость в оси y,

cmd_vel.angular.z – угловая скорость в оси z) и преобразует их в команды на моторы мобильной базы робота.

В следующих подглавах будут подробно описаны этапы создания и настройки вышеперечисленных компонентов для создания системы навигации.

6.2 Преобразование сигналов системы навигации в сигналы управления робомобилем

Следующим этапом работы стала совместная организация имеющихся нод управления движением модели с темой cmd_vel, с которой взаимодействует модуль навигации move_base. На данном этапе был взята готовая шаблонная реализация cmd_vel с официального сайта ROS [50] для роботов с геометрией Акерманна. Для того, чтобы связать два отдельных модуля, потребовалось создать единую тему сообщений /junior_car/ackermann_cmd, которая объединяла данные steer_remap и drive_remap в единое сообщение типа ackermann_msgs/AckermannDriveStamped.

6.3 Корректировка данных с лазера модели робота

Для корректной работы системы навигации одним из важных требований аппаратного обеспечения робота является правильно установленный лазер (лазер, расположенный на поверхности, параллельной плоскости земли). В случае мобильного робота «Юниор», лазерный дальномер на нем располагается под углом ~15 градусов. Так как модель повторяет физические характеристики реального робота, созданный лазер в Gazebo также имеет этот угол наклона. Однако это противоречит требованиям исправной работы системы навигации. Для устранения этого противоречия было принято решение создать ноду rectified_scan, которая бы считывала поступающие данные с лазера модели и

корректировала их в данные, которые соответствуют правильно установленному лазеру (устраняя существующий угол наклона).

Нода rectified_scan слушает тему /scan, в которой публикуются сообщения, отправляемые лазером модели робота. При получении нового сообщения от лазера, нода создает свое собственное сообщение типа sensor_msgs/LaserScan, корректируя следующие поля полученного сообщения: std_msgs/Header header, float32[] ranges, float32[] intensities. В свою очередь, в Header заменяется поле frame_id, указывая на новую ось виртуального лазера, расположенного согласно требованиям системы навигации; в полях float32[] ranges, float32[] intensities происходит корректировка входящих значений от данных сообщений темы /scan. Новое сообщение с данными публикуется в другой теме, которая далее использует метод одновременной локализации и картографирования hector_mapping. Исходный код пакета корректировки данных с лазера представлен в Приложении В.

6.4 Конфигурация локального планировщика

Конфигурация локального планировщика хранится в файле teb_local_planner.yaml (см. Приложение Г.4). Конфигурация представляет собой перечисление параметров локального планировщика и их значение в случае конкретного робота. Каждый из параметров имеет свои значения по умолчанию, которые могут подходить для всех роботов, однако рекомендуется уточнять все параметры, опираясь на свойства конкретного робота. Содержание всех файлов с параметрами, описанными далее, могут быть изучены в Приложении Г.4.

В случае неголономных роботов, имеющих рулевое управление (передние колеса робота могут поворачиваться и задавать направление движения робота) в ROS существует небольшое количество планировщиков (глобальных и локальных), которые бы учитывали специфичную конфигурацию робота. В

общем случае в стеке навигации робот является голономным (всенаправленным). Примером доступного (открытого) локального планировщика для роботов с геометрией Аккермана является Teb Local Planner [51]. Пакет teb_local_planner реализует плагин для base_local_planner, 2D системы навигации. Лежащий в основе планировщика метод Timed Elastic Band [52] [53], он локально оптимизирует траекторию робота с учетом времени выполнения траектории, избегания препятствий, соблюдая заданную максимальную скорость и ускорение. Данный подход основан на другом существующем методе, Elastic Band, имеющем свою реализацию в виде локального планировщика в ROS (eband_local_planner). Подход Elastic Band был создан ученым Оуссама Хатиб (Oussama Khatib) и опубликован в 1993 году [54].

Teb_local_planner публикует сообщений в ряд тем: global_plan, local_plan, teb_poses, teb_markers, teb_feedback.

- global_plan (тип сообщения nav_msgs/Path) – глобальный план, которому локальный планировщик в настоящее время пытается следовать. Используется в основном для визуализации.
- local_plan (тип сообщения nav_msgs/Path) – локальный план или траектория, которую оптимизирует teb_local_planner и следует ей. Используется в основном для визуализации.
- teb_poses (тип сообщения geometry_msgs/PoseArray) – список дискретных поз (SE2 пространства) текущего локального плана. Используется в основном для визуализации.
- teb_markers (тип сообщения visualization_msgs/Marker) – teb_local_planner предоставляет дополнительную информацию о сцене планирования через маркеры с различным пространством имен. Пространства имен PointObstacles и PolyObstacles визуализируют все точечные и полигональные препятствия, которые в данный момент учитываются при оптимизации. Пространство имен TebContainer визуализирует все найденные и оптимизированные траектории, опирающиеся на

альтернативные топологии (только если включено параллельное планирование).

- teb_feedback (тип сообщения teb_local_planner/FeedbackMsg) – сообщение обратной связи содержит запланированную траекторию, включая профиль скорости, временную информацию, список препятствий. Используется для отладки. Для работы этой темы необходимо включить параметр «publish_feedback».

В свою очередь, локальный планировщик подписан на следующий ряд тем: odom, obstacles, via_points.

- odom (тип сообщения nav_msgs/Odometry) – информация об одометрии, которая дает локальному планировщику текущую скорость робота.
- obstacles (тип сообщения costmap_converter/ObstacleArrayMsg) – предоставление пользовательских препятствий в виде точечных, линейных или многоугольных препятствий (в дополнение к препятствиям карты затрат или вместо них).
- via_points (тип сообщения nav_msgs/Path) – предоставляет пользовательские промежуточные точки.

Параметры teb_local_planner позволяют пользователю настраивать поведение планировщика и, вследствие этого, робота. Параметры сгруппированы в несколько категорий: конфигурация робота, точность конечной позиции (англ. goal tolerance), конфигурация траектории, препятствий, оптимизация, планирование в особых топологиях и другие (прочие) параметры. В yaml файле конфигурации содержится перечисление этих параметров с определенным значением (заданным пользователем). В случае, если параметр не задан в файле, планировщик использует значения по умолчанию для любого параметра.

6.4.1 Конфигурация робота

Параметры конфигурации робота представлены в Таблице 6.4.1.1, Приложении Г.4. В первом столбце перечислены все параметры, во втором дано краткое пояснение, в третьем – их значение по умолчанию, в четвертом – определенное значение для модели робота «Юниор».

Таблица 6.4.1.1 – параметры конфигурации робота

Наименование параметра	Пояснение	Значение по умолчанию	Заданное значение
acc_lim_x	максимальное поступательное ускорение робота в м/с^2	0.5	10
acc_lim_theta	максимальное угловое ускорение робота в рад/с^2	0.5	10
max_vel_x	максимальная поступательная скорость робота в м/с	0.4	10
max_vel_x_backwadrs	максимальная абсолютная поступательная скорость робота при движении назад в м/с	0.2	4
max_vel_theta	максимальная угловая скорость робота в рад/с	0.3	10
min_turning_radius	минимальный радиус поворота car-like робота	0.0	1.0
wheelbase	дистанция между задней и передней осями колес	1.0	1.0
max_vel_y	максимальная скорость движения робота (должна быть 0 для неголономных роботов)	0.0	0.0
acc_lim_y	максимальное ускорение робота	0.5	10
footprint_model/type	тип модели проекции робота на плоскость, используемый для оптимизации. регламентированные типы: point, circular, line, two_circles, polygon	point	polygon
footprint_model/line_start	этот параметр актуален только для типа «line». Содержит начальные координаты отрезка.	[-0.3,0.0]	-
footprint_model/line_end	этот параметр актуален только для типа «line». Он содержит конечные координаты отрезка.	[0.3,0.0]	-
footprint_model/front_offset	этот параметр актуален только для типа "two_circles". Он описывает, насколько центр переднего круга смещен вдоль оси X робота. Предполагается, что ось вращения робота находится в [0,0].	0.2	-
footprint_model/front_radius	этот параметр актуален только для типа "two_circles". Содержит радиус переднего круга.	0.2	-

Продолжение таблицы 6.4.1.1

Наименование параметра	Пояснение	Значение по умолчанию	Заданное значение
footprint_model/rear_offset	этот параметр актуален только для типа "two_circles". Он описывает, насколько центр заднего круга смещен вдоль отрицательной оси X робота. Предполагается, что ось вращения робота находится в [0,0].	0.2	-
footprint_model/rear_radius	этот параметр актуален только для типа "two_circles". Содержит радиус заднего круга.	0.2	-
footprint_model/vertices	этот параметр актуален только для типа «polygon». Он содержит список вершин многоугольника (2D координаты). Многоугольник всегда замкнут.	[[0.25,-0.05],[…],…]	[[-0.556, -0.325],[-0.556, 0.325],[0.556, 0.325],[0.556, -0.325]]

6.4.2 Точность конечной позиции

Точность конечной позиции определяет числовые значения допущения отклонения от требуемой конечной позиции робота. Допущения выражаются через два основных значения: допустимое евклидово расстояние от целевой точки (в метрах) и допустимая ошибка ориентации робота (в радианах). Параметры этой категории представлены в Таблице 6.4.2.1, Приложении Г.4.

Таблица 6.4.2.1 – параметры точности конечной позиции

Наименование параметра	Пояснение	Значение по умолчанию	Заданное значение
xy_goal_tolerance	допустимое конечное евклидово расстояние до цели в метрах	0.2	0.2
yaw_goal_tolerance	допустимая конечная ошибка ориентации в радианах	0.2	0.3
free_goal_vel	днятие ограничения скорости цели, чтобы робот мог достичь цели с максимальной скоростью	false	false

6.4.3 Конфигурация траектории

Параметры категории представлены в Таблице 6.4.3.1, Приложении Г.4.

Таблица 6.4.3.1 – параметры конфигурации траектории

Наименование параметра	Пояснение	Значение по умолчанию	Заданное значение
dt_ref	требуемое временное разрешение траектории (траектория будет изменяться между итерациями, если dt_ref + -dt_hysteresis нарушается)	0.3	0.6
dt_hysteresis	гистерезис для автоматического изменения размера траектории в зависимости от текущего временного разрешения	0.1	0.1
min_samples	минимальное количество образцов (должно быть больше 2)	3	3
global_plan_overwrite_orientation	перезапись ориентации локальных подцелей, предоставляемых глобальным планировщиком	true	true
global_plan_viapoint_sep	если значение параметра положительное, промежуточные точки удаляются из глобального плана	-0.1 (отключен)	-0.1
max_global_plan_lookahead_dist	указание максимальной длины (совокупного евклидово расстояния) подмножества глобального плана, учитываемого для оптимизации	3.0	3.0
force_reinit_new_goal_dist	повторная инициализация траектории, если предыдущая цель обновлена с разделением больше указанного значения в метрах	1.0	1.0
feasibility_check_no_poses	указание, до какой позиции в прогнозируемом плане следует проверять выполнимость каждого интервала выборки	4	0
publish_feedback	публикация обратной реакции планировщика, содержащая полную траекторию и список активных препятствий (должен быть включен только для оценки или отладки)	false	false
shrink_horizon_backup	позволяет планировщику временно уменьшить горизонт (50%) в случае автоматически обнаруженных проблем (например, невыполнимость решения)	true	true

Продолжение таблицы 6.4.3.1

Наименование параметра	Пояснение	Значение по умолчанию	Заданное значение
allow_init_with_backwards_motion	если значение Истины, базовые траектории могут быть инициализированы с движениями назад, если цель находится за начальной точкой в локальной карте затрат (рекомендуется, только если робот оснащен задними датчиками)	false	false
exact_arc_length	если значение Истины, планировщик использует точную длину дуги в расчетах скорости, ускорения и скорости поворота, в противном случае используется евклидово приближение	false	false
shrink_horizon_min_duration	указание минимальной продолжительности для уменьшенного горизонта в случае обнаружения невыполнимой траектории	10.0	10.0

6.4.4 Конфигурация препятствий

Конфигурация препятствий описывает поведение планировщика в отношении препятствий на карте. Кроме стандартного параметра минимального расстояния до препятствия, teb_local_planner учитывает специфичные для него параметры, также представленные в Таблице 6.4.4.1, Приложении Г.4.

Таблица 6.4.4.1 – параметры конфигурации препятствий

Наименование параметра	Пояснение	Значение по умолчанию	Заданное значение
min_obstacle_dist	минимальное желаемое расстояние до препятствий	0.5	0.7
include_costmap_obstacles	указание учета препятствия в локальной карте затрат. Каждая ячейка, помеченная как препятствие, рассматривается как точка-препятствие.	true	true
costmap_obstacles_behind_robot_dist	ограничение занятых локальных препятствий карты затрат, принятые во внимание для планирования позади робота (указание расстояния в метрах)	1.0	1.5

Продолжение таблицы 6.4.4.1

Наименование параметра	Пояснение	Значение по умолчанию	Заданное значение
obstacle_poses_affected	каждая позиция препятствия «прикрепляется» к ближайшей позиции на траектории, чтобы сохранить дистанцию. Дополнительные соседи также могут быть приняты во внимание	30	30
inflation_dist	зона буфера вокруг препятствий с ненулевой штрафной стоимостью (должна быть больше, чем min_obstacle_dist для вступления параметра в силу)	0.6	0.6
include_dynamic_obstacles	если для этого параметра установлено значение Истины (true), движение препятствий с ненулевой скоростью прогнозируется и учитывается при оптимизации с помощью модели с постоянной скоростью	false	false
legacy_obstacle_association	стратегия соединения позиции траекторий с препятствиями для оптимизации была изменена. Возможен выбор старой / предыдущей стратегии, установив для этого параметра значение true	false	false
obstacle_association_cutoff_factor	этот параметр используется только в том случае, если для параметра legacy_obstacle_association задано значение false	5	5
obstacle_ association_force_inclusion_factor	стратегия ассоциации не преемственных препятствий используется во время оптимизации для соединения к дискредитированной траектории только значимых препятствий	1.5	1.5

6.4.5 Параметры оптимизации

Параметры оптимизации является расширением стандартного подхода Elastic Band, описанные в работах [55] [56]. Параметры оптимизации представлены в Таблице 6.4.5.1, Приложении Г.4.

Таблица 6.4.5.1 – параметры оптимизации

Наименование параметра	Пояснение	Значение по умолчанию	Заданное значение
no_inner_iterations	количество фактических решающих итераций, вызываемых в каждой внешней итерации	5	5
no_outer_iterations	каждая внешняя итерация автоматически изменяет размер траектории в соответствии с требуемым временным разрешением dt_ref и вызывает внутренний оптимизатор	4	4
penalty_epsilon	добавить небольшой запас к штрафным функциям для приближений с жесткими ограничениями	0.1	0.1
weight_max_vel_x	вес оптимизации для удовлетворения максимально допустимой поступательной скорости	2.0	20
weight_max_vel_theta	вес оптимизации для удовлетворения максимально допустимой угловой скорости	1.0	20
weight_kinematics_nh	вес оптимизации для удовлетворения неголономной кинематики	1000.0	1000.0
weight_kinematics_forward_drive	вес оптимизации для того, чтобы заставить робота выбирать только прямые направления (положительная поступательная скорость). Небольшой вес (1.0) по-прежнему позволяет двигаться роботу назад	1.0	53
weight_kinematics_turning_radius	вес оптимизации для обеспечения минимального радиуса поворота (только для car-like роботов)	1.0	5
weight_optimaltime	вес оптимизации для сокращения траектории с переходом / временем выполнения	1.0	1.0

6.4.6 Конфигурация планирования в особых топологиях

Параметры конфигурации в особых топологиях также являются расширением стандартного подхода Elastic Band. Параметры описаны в Таблице 6.4.6.1, Приложении Г.4.

Таблица 6.4.6.1 – параметры планирования в особых топологиях

Наименование параметра	Пояснение	Значение по умолчанию	Заданное значение
enable_homotopy_class_planning	активация параллельного планирования в отличительных топологиях	true	false
enable_multithreading	активация нескольких потоков, чтобы планировать каждую траекторию в отдельном потоке	true	true
max_number_classes	максимальное количество учитываемых траекторий (ограничивает вычислительные усилия)	4	4
selection_cost_hysteresis	какую стоимость траектории должен иметь новый кандидат в ранее выбранной траектории для выбора (выбор, если new_cost <old_cost *factor)	1.0	1.0
selection_obst_cost_scale	дополнительное масштабирование условий стоимости препятствий только для выбора «лучшего» кандидата	100.0	100.0
selection_viapoint_cost_scale	дополнительное масштабирование с точки зрения стоимости через точки только для выбора «лучшего» кандидата	1.0	1.0
selection_alternative_time_cost	если это правда, стоимость времени (сумма квадратов временных разностей) заменяется общим временем перехода (сумма временных разниц).	false	1.0
roadmap_graph_no_samples	указание количества образцов, сгенерированных для создания графика дорожной карты	15	15
roadmap_graph_area_width	случайные ключевые точки / путевые точки отбираются в прямоугольной области между началом и целью. Указывается ширина этого региона в метрах	6	5

Продолжение таблицы 6.4.6.1

Наименование параметра	Пояснение	Значение по умолчанию	Заданное значение
h_signature_prescaler	внутренний параметр масштаба (H-подпись), который используется для различения гомотопических классов	1.0	0.5
h_signature_threshold	активация параллельного планирования в отличительных топологиях	0.1	0.1
obstacle_heading_threshold	указание значение скалярного произведения между началом препятствия и целью, чтобы учесть их (препятствия) для исследования	1.0	0.45
visualize_hc_graph	визуализируйте график, который создан для исследования отличительных траекторий	false	false
viapoints_all_candidates	если значение Истины, все траектории различных топологий привязаны к набору промежуточных точек, в противном случае с ними связана только траектория, имеющая ту же топологию, что и первоначальный / глобальный план (не влияет на test_optim_node)	true	true
switching_blocking_period	указание длительности в секундах, которая должна истечь, прежде чем будет разрешен переход на новый класс эквивалентности	0.0	0.0

6.4.7 Другие параметры

Конфигурация других параметров включает в себя описание темы одометрии робота и наименование глобальной системы координат. Параметры представлены в Таблице 6.4.7.1, Приложении Г.4.

Таблица 6.4.7.1 – другие параметры планировщика

Наименование параметра	Пояснение	Значение по умолчанию	Заданное значение
map_frame	система координат глобального планирования (в случае статической карты этот параметр обычно должен быть изменен на "/ map")	odom	/map

Продолжение таблицы 6.4.7.1

Наименование параметра	Пояснение	Значение по умолчанию	Заданное значение
odom_topic	название темы одометрии	odom	odom

6.5 Конфигурация глобального планировщика

Конфигурация глобального планировщика описана в файле avrora_unior_nav.yaml. В качестве глобального планировщика был выбран SBPLLatticePlanner. Пакет sbpl_lattice_planner является оболочкой SBPL Lattice окружения, соответствующий интерфейсу nav_core::BaseGlobalPlanner, указанному в nav_core [57]. Благодаря этому планировщик lattice_planner может быть использован как глобальный планировщик в move_base.

Планировщик генерирует путь от текущей позиции робота до желаемой цели. Путь генерируется путем объединения серий «примитивов движения» (англ. motion primitives), которые представляют собой короткие, кинематические возможные движения робота. Таким образом, планирование выполняется в измерениях x, y и theta, что приводит к плавным траекториям пути, учитывающим ориентацию робота. Это является ключевым фактором для роботов не круглой формы или роботов, имеющих неголономные ограничения.

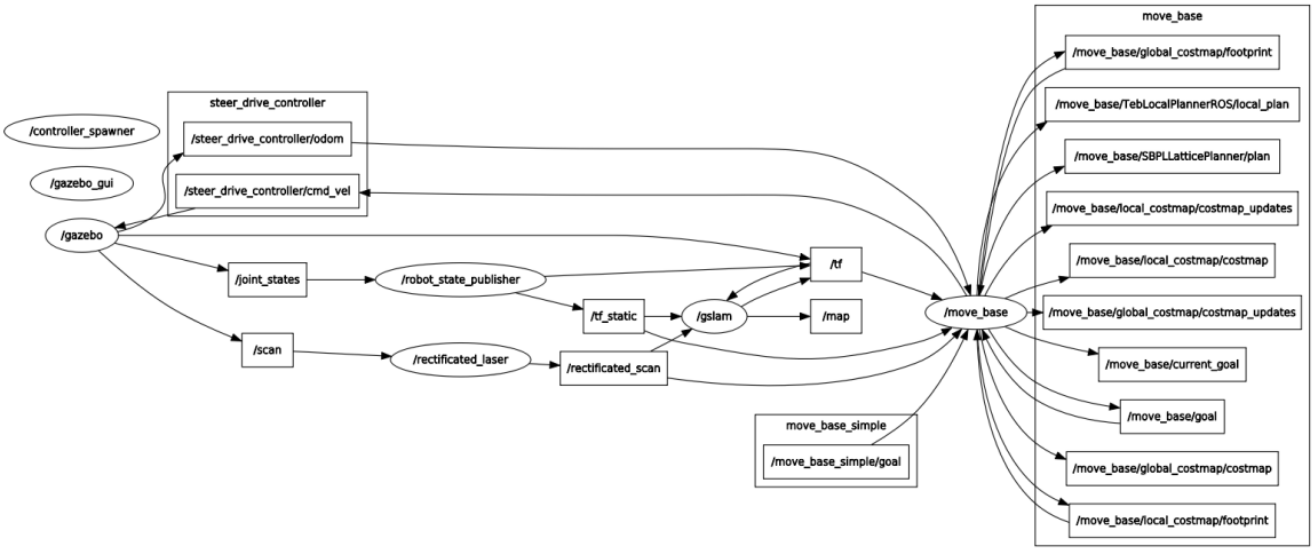

Рисунок 6.5.1 – Визуализация работы стека навигации в rqt_graph

В случае с робомобилем «Юниор», робот имеет неголономные ограничения и не совместим с большинством других планировщиков в ROS. Визуализация работы системы навигации представлена на Рисунке 6.5.1.

6.6 Конфигурация карт стоимости планировщика

Система навигации использует две карты стоимости (англ. costmap) для хранения информации о препятствиях в окружающем мире. Одна карта стоимости используется для глобального планирования, для создания долгосрочных планов по всей окружающей среде; другая карта используется для локального планирования и избегания препятствий. Каждая карта стоимости имеет ряд параметров, которые необходимо задать для конкретного робота и системы навигации. Принято разделять параметры на три типа: общие параметры конфигурации, параметры глобальной конфигурации, параметры локальной конфигурации.

6.6.1 Понятие карты стоимости

Costmap_2d является пакетом в ROS, который предоставляет настраиваемую структуру (содержащую информацию о том, куда может перемещаться робот) в виде сетки занятости (англ. occupancy grid). Карта стоимости использует данные датчиков и информацию со статической карты для хранения и обновления информации о препятствиях в мире с помощью объекта costmap_2d::Costmap2DROS.

Карта стоимости автоматически подписывается на темы датчиков через ROS и обновляется соответствующим образом. Каждый датчик используется для маркировки (добавления информации о препятствиях в карту стоимости), удаления (удаления информации о препятствиях из карты стоимости) или для

обеих целей. Операция маркировки определена для изменений стоимости ячейки в виде индекса массива карты. Операция очистки выполняет перерасчет карты используя трассировку набора лучей, построенных из точки где находится датчик через каждую точку где были ранее зафиксированы препятствия. Если для описания препятствий используются трехмерные структуры, информация о препятствии для каждой из вертикальных колонок проецируется вниз для получения двумерной карты стоимости.

Хотя каждая ячейка в карте стоимости имеет значение в интервале от 0 до 255 различных значений стоимостей, используемая базовая структура (сетка занятости) имеет только три стоимости. Каждая ячейка в этой структуре может быть свободной, занятой или неизвестной. Каждому статусу присваивается специальное значение при проекции на карту стоимости.

Карта стоимости выполняет циклы обновления карты с частотой, указанной в параметре update_frequency. В каждом цикле поступают данные с датчиков, операции маркировки и удаления выполняются в базовой структуре (сетке занятости) карты стоимости, которая затем проецируется на карту стоимости с соответствующими значениями.

Инфляция – процесс распространения значений стоимости из занятых ячеек, уменьшающихся с увеличением расстояния от препятствия (см. Рисунок 6.6.1.1). Инфляция необходима для увеличения расстояний от робота до препятствий, тем самым обеспечивая безопасность робота во время навигации по карте.

В этом процессе выделяют 5 конкретных стоимостей для значений карты стоимости.

- «Летальная» стоимость (англ. lethal cost) означает, что в данной ячейке существует реальное препятствие. Поэтому, если бы центр робота находился в этой ячейке, робот бы не избежал столкновения.
- «Вписанная» стоимость (англ. inscribed cost). Под вписанной стоимостью принимается стоимость ячейки если она находится от настоящего

препятствия на расстоянии меньше чем радиус окружности, вписанной в след (англ. footprint) робота.

- «Возможная вписанная» стоимость (англ. possibly circumscribed cost). Под "возможно-описанной" стоимостью понимается стоимость, определенная по аналогии с вписанной стоимостью, с заменой вписанной окружности на описанную. Такая стоимость говорит о том, что если центр робота находится в ячейке с равным или больше значением, то опасность столкновения с ней будет зависеть от ориентации робота. В данном случае используется термин "возможно-", так как такая стоимость определяет напрямую препятствие, конкретные значения определяются пользователем по своему предпочтению. Например, пользователь может обусловить избегание роботом определенных частей здания путем задания значений для таких областей на карте стоимости.
- «Свободная» стоимость (англ. freespace cost) предполагает, что стоимость данной ячейки равна нулю; это означает, что нет ничего, что препятствует роботу попасть с эту ячейку.
- «Неизвестная» стоимость (англ. unknown cost) означает отсутствие информации о данной ячейке.

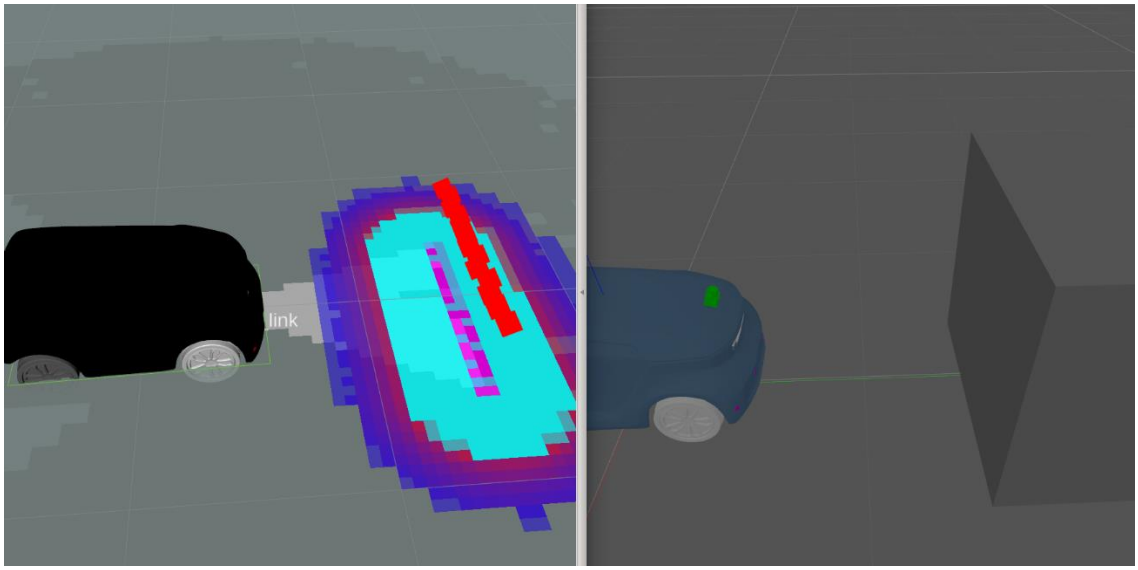

Рисунок 6.6.1.1- отображение локальной карты стоимости (слева) на примере препятствия «куб»

Всем остальным стоимостям присваивается значение между «свободным пространством» и «возможно ограниченным», в зависимости от их расстояния от «летальной» ячейки.

6.6.2 Общая конфигурация карт стоимостей

Система навигации использует карты затрат для хранения информации о препятствиях в мире. Для того, чтобы информация была актуальной и верной, необходимо указать темы («шины» для передачи определенного типа данных в ROS) для карт стоимости, сообщения которых они должны анализировать для обновления карт. Общая конфигурация карт описана в файле common_costmap.yaml (полное содержание файла common_costmap.yaml представлено в приложении Г.5). Ниже приведено описание некоторых параметров и их значение (см. Таблица 6.6.2.1).

Таблица 6.6.2.1 – общая конфигурация карт стоимостей

Наименование параметра	Пояснение	Значение
footprint	Проекция робота на плоскость	[[-0.556, -0.325],[-0.556, 0.325],[0.556, 0.325],[0.556, -0.325]]
footprint_padding	Дополнительное расстояние между проекцией робота на плоскость и препятствием	0.0
transform_tolerance	Определение задержки в преобразовании (tf) данных, которая допустима в секундах. Этот параметр служит гарантией потери ссылки в дереве tf, в то же время оставляя время ожидания, достаточное для пользователя	0.1
robot_base_frame	Определение системы координат базы робота, на которую должна ориентироваться карта затрат	footprint
global_frame	Определение, в какой координатной системе должна работать карта затрат	map
update_frequency	Определение частоты в Гц, с которой карта затрат будет запускать свой цикл обновления	5.0
publish_frequency	Определение частоты в Гц, с которой карта затрат будет публиковать информацию визуализации	1.0

Продолжение таблицы 6.6.2.1

Наименование параметра	Пояснение	Значение
rolling_window	Установка параметра в значении Истины означает, что карта затрат будет оставаться центрированной вокруг робота, пока робот движется по миру	true
resolution	Разрешение (в метр / ячейка) карты затрат	0.05
obstacles_laser	Структура для описания датчиков робота, которые необходимы для построения и обновления карты затрат. Параметр «selection_sources» определяет список датчиков, которые будут передавать информацию в карту затрат, разделенных пробелами	observation_sources: laser laser data_type: LaserScan clearing: true marking: true topic: rectificated_scan inf_is_valid: true
inflation layer	Описание структуры инфляции. Радиус инфляции должен быть установлен на максимальном расстояние от препятствий, при которых должны быть понесены расходы	inflation_radius: 0.7 enabled: true

6.6.3 Конфигурация глобальной карты стоимости

Файл global_costmap.yaml содержит параметры, специфичные для глобальной карты стоимостей. Их определение представлено в Таблице 6.6.3.1, Приложении Г.7.

Таблица 6.6.3.1 – конфигурация глобальной карты стоимости

Наименование параметра	Пояснение	Значение
resolution	Разрешение (в метр / ячейка) карты затрат	0.10
update_frequency	Определение частоты в Гц, с которой карта затрат будет запускать свой цикл обновления	5.0
publish_frequency	Дополнительное расстояние между Определение частоты в Гц, с которой карта затрат будет публиковать информацию визуализации	5.0

6.6.4 Конфигурация локальной карты стоимости

Файл local_costmap.yaml содержит параметры, специфичные для локальной карты стоимостей. Их определение представлено в Таблице 6.6.4.1, Приложении Г.6.

Таблица 6.6.4.1 – конфигурация локальной карты стоимости

Наименование параметра	Пояснение	Значение
static_map	Параметр «static_map» определяет, должна ли инициализироваться карта затрат на основе карты, обслуживаемой map_server	false
width	Ширина карты в метрах	6.0
height	Высота карты в метрах	6.0
resolution	Разрешение (в метр / ячейка) карты затрат	0.05

6.7 Апробация системы навигации в Rviz

Для апробации системы навигации в Rviz была разработана следующая процедура.

1. Проверка навигации робота в пустом мире. Выбор цели определяется таким образом, чтобы путь прокладывался строго прямо, с минимальным изменением ориентации робота в конечной (целевой) точке (см. Рисунок 6.7.1).
2. Проверка навигации робота в пустом мире. Выбор цели определяется таким образом, чтобы целевая точка была в ориентации, отличной от первоначальной ориентации робота.
3. Проверка навигации робота в мире с препятствиями (см. Рисунок 6.7.2). Цель выбирается таким образом, чтобы путь к ней выстраивался с учетом обхода препятствия.

Эксперименты проводились согласно процедурам. Для апробации потребовались следующие шаги:

- Запуск модели робота в симуляторе Gazebo
- Запуск контроллеров модели
- Запуск системы навигации
- Запуск Rviz с файлом конфигурации, описанным в Приложениях Г.8, Г.9

- Отладка процедур проводилась при отслеживании сообщений, отправляемых на темы контроллеров, move_base (см. Рисунок 6.7.3).

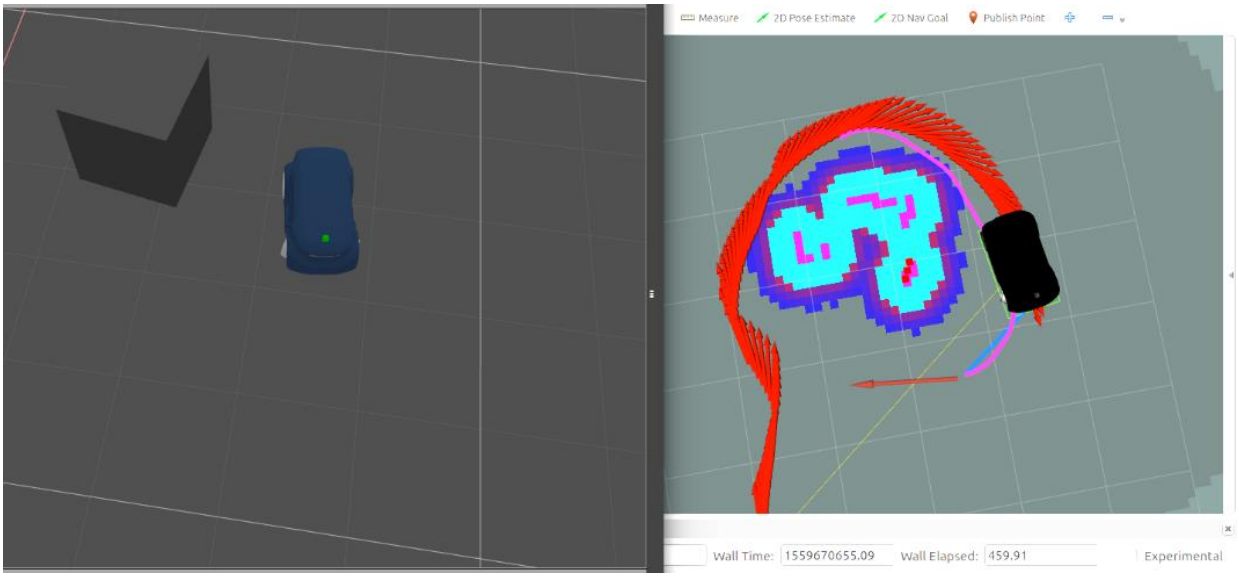

Рисунок 6.7.1 – Навигация робота в мире с препятствиями: визуализация симуляции (слева) и данных робота через Rviz (справа)

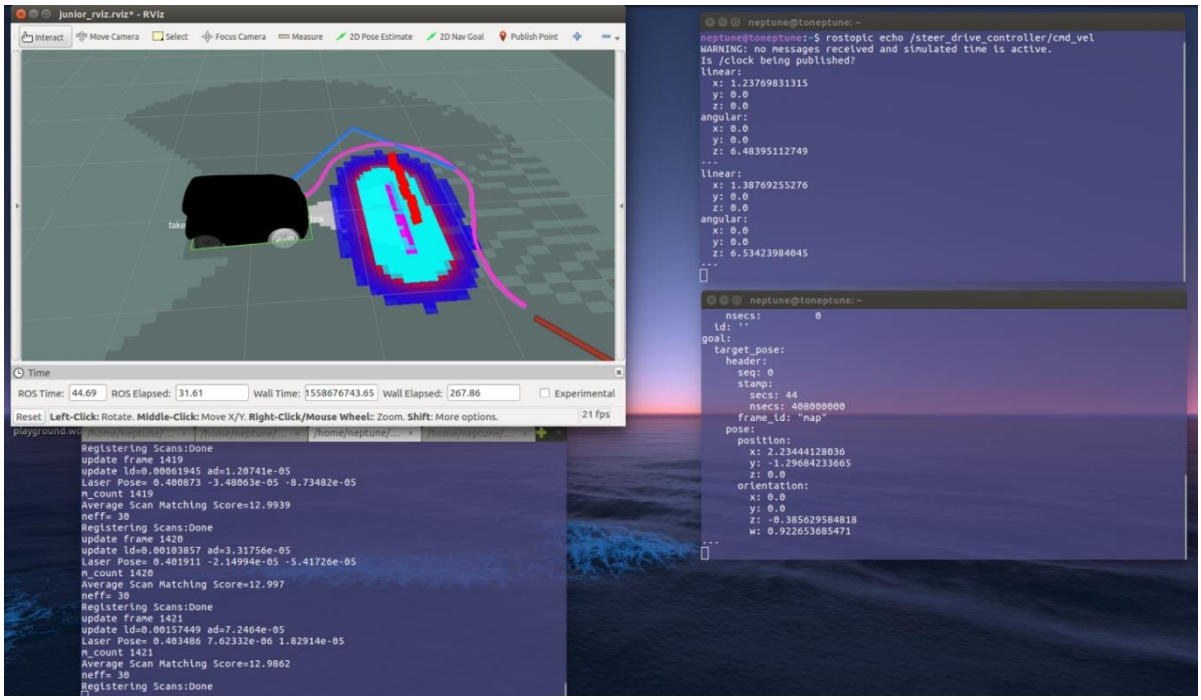

Рисунок 6.7.3 – Отладка стека навигации

7 КОНЦЕПЦИЯ РЕШЕНИЯ КЕЙСА

Концепция решения кейса состоит из двух этапов. Первая часть – подготовка к апробации. Она включает в себя: моделирование мира в симуляторе Gazebo, соответствующего городской среде; созданию ноды, которая будет отправлять в Система навигации робота целевые точки. Вторая часть – апробация. Процедура апробации следующая: в запущенном виртуальном мире робот будет совершать автономную навигацию в точки доставок; по достижении роботом поставленной целевой точки, нода будет отправлять на Система навигации следующую точку.

7.1 Создание мира в Gazebo

Для апробации автономной навигации модели в точки доставок необходимо создать соответствующий мир в Gazebo, топология которого была бы схожей с городской средой. Для создания мира использовались стандартные 3D модели, доступные в базе моделей симулятора Gazebo. На Рисунках 7.1.1, 7.1.2 представлен созданный мир в Gazebo.

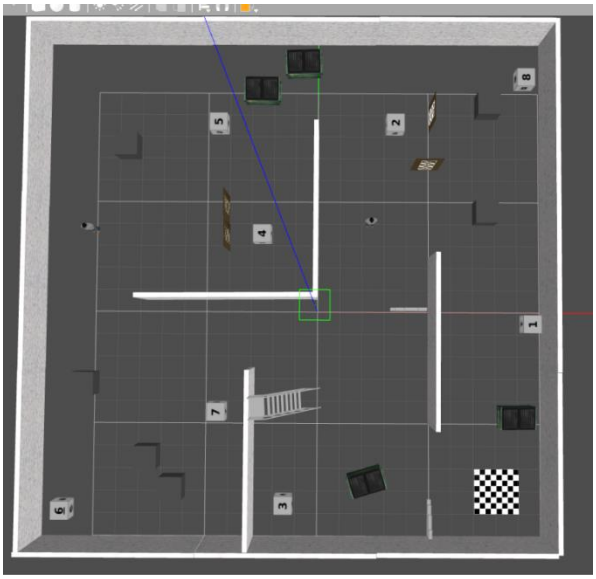

Рисунок 7.1.1 – Разработанный мир в Gazebo

В мире находятся объекты (кубы), пронумерованные от 1 до 9, которые являются тестовыми точками доставок (см. Рисунок 7.1.3). Размер мира 23 метра в длину и 23 метра в ширину. Внутри мира добавлены панели для имитации городских зданий. Ширина проемов равна 3 метрам, что соответствует ширине дороги с двумя дорожными полосами. В мир также произвольным образом добавлены препятствия различной формы и габаритов (см. Рисунок 7.1.3) для увеличения сложности построения маршрута.

7.2 Апробация автономной навигации робота в точки доставок

В качестве апробации идеи последнего километра использовался мир, описанный в главе 7.1 и Система навигации модели робота. В Gazebo загружались модель робота и мир с городской средой. Далее запускалась Система навигации и выбирались целевые точки для их достижения. Система навигации выстраивал путь, учитывая обнаруживаемые роботом препятствия. Примеры апробации в созданной среде представлены на Рисунках 7.2.1.

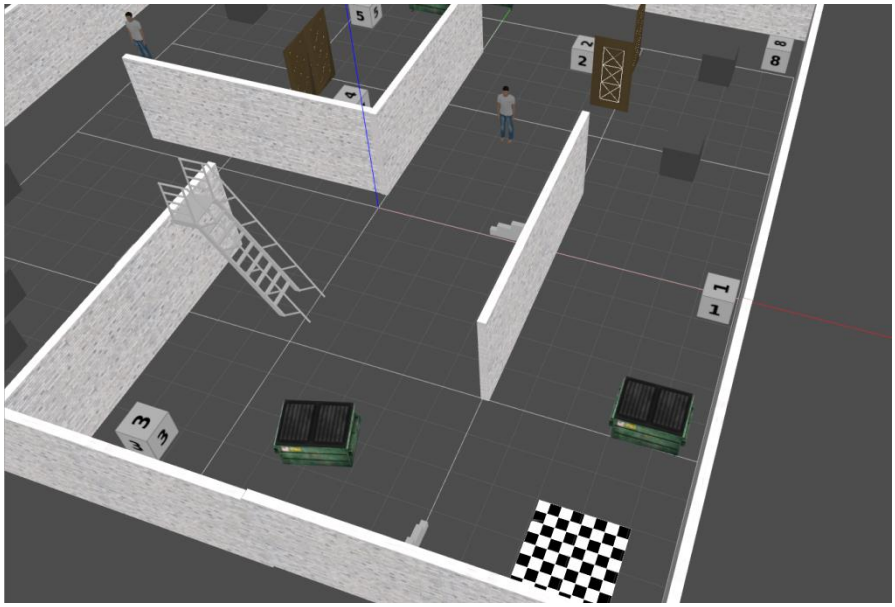

Рисунок 7.1.2 – Пример среды с разнородными препятствиями

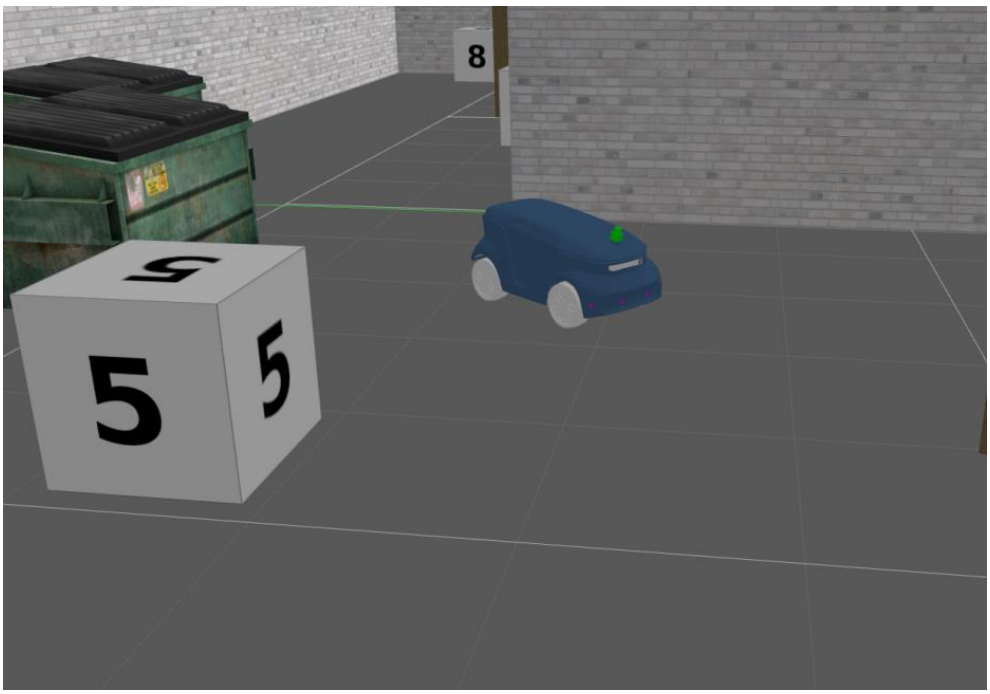

Рисунок 7.1.3 - Куб доставки под номером 5

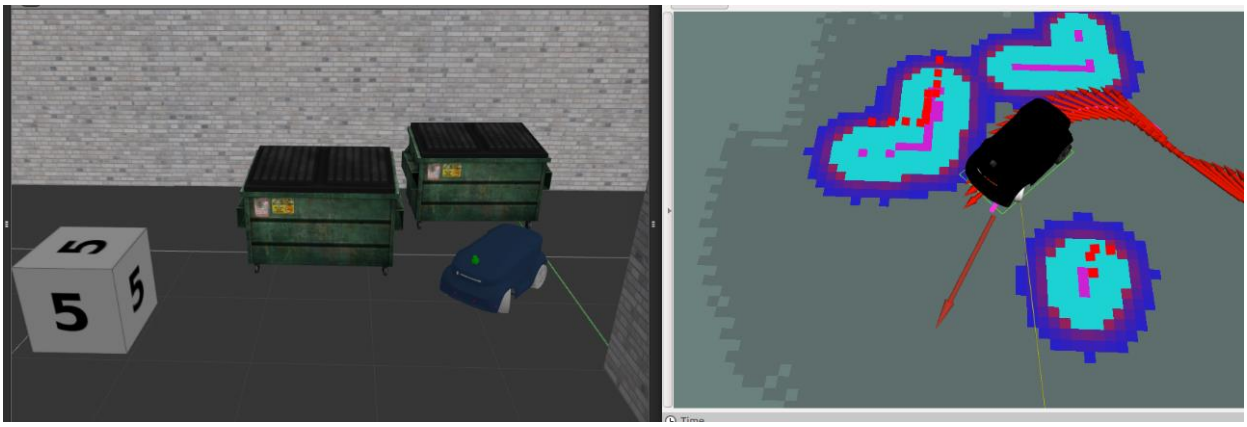

Рисунок 7.2.1 – Апробация кейса последнего километра в симуляции

8 ДАЛЬНЕЙШИЕ ПЛАНИРУЕМЫЕ ИССЛЕДОВАНИЯ

После апробации идеи решения кейса в симуляции, возможным продолжением исследования является подготовка мобильного робота «Юниор» к экспериментам в реальной среде с последующим их проведением. Под подготовкой подразумевается ряд работ, связанных с имплементацией разработанных для симуляционной модели пакетов и различных настроек на реальном роботе. Особенность данной работы состоит в том, что в программном обеспечении робота (ROS) находятся только базовые пакета для работы с датчиками робота и управлением его моторами. Возможная перспектива развития проекта следующая.

- Разработка системы управления движением робомобиля
- Имплементация разработанных пакетов на робомобиле
- Настройка системы навигации для робомобиля
- Апробация системы навигации в лабораторных условиях
- Создание искусственной городской среды для апробации задачи последнего километра
- Апробация робомобиля для решения задачи последнего километра в искусственных городских условиях.

8.1 Разработка системы управления движением робомобиля

Разработка системы управления движением робомобиля содержит в себе разработку нод для контроля робомобиля. Данный этап необходим, так как текущий пакет контроллера робомобиля ur_hardware_driver не предоставляет точное управление робомобилем. Тема ноды ur_hardware_driver принимает сообщения типа std_msgs/Float64; данные сообщения носят абстрактный характер. В случае, если пользователь отправляет сообщения на задние моторы робота, то необходимо подбирать значения из промежутка [6.0; 100.0]; при этом,

пользователь должен самостоятельно определить, как увеличение значения посылаемого сообщения влияет на увеличение образующейся скорости робомобиля. В случае, если пользователь отправляет сообщения на мотор рулевой рейки, промежуток значений, который поворачивает на определенный угол рулевую систему робота приблизительно составляет [20.0;56.0]. в этой ситуации пользователь также не может предположить, какое точное значение величины обеспечит поворот рулевой системы робота на нужный угол.

Есть два похода к реализации данного этапа. Первый подход заключается в том, чтобы создать новый контроллер управления моторами робомобиля, заменяя пакет, предоставленный компанией-производителем. Второй подход состоит в разработке пакета-адаптера, нода которого бы принимала от пользователя (или системы навигации) четкие команды скорости (скорость в м/с) и угла поворота колес (угол в радианах), и конвертировала их в нужные значения для пакета ur_hardware_driver.

8.2 Имплементация ROS пакетов системы навигации

Данный раздел объединяет в себе сразу два этапа, так как они оба заключаются в настройке робомобиля на базе пакетов и конфигураций его модели. Разработанный для модели пакет корректировки данных с лазерного дальномера должен быть аналогично применен для корректировки данных реального датчика.

Следующей задачей является настройка параметров модели для реалистичности представления робота «Юниор» в среде Gazebo, так как в задачах автономной навигации реальный робот будет оценивать свое положение, восприятие мира через виртуальную модель. Эта задача является необходимой для любых будущих исследований, связанных с робомобилем.

После настройки системы навигации на реальном роботе его апробация может быть проведена в лабораторных условиях. Под лабораторными условиями

подразумеваются офисные кабинеты и коридор учебного корпуса, площадь которых позволяет робомобилю свободно передвигаться и совершать развороты.

8.3 Апробация задачи последнего километра в искусственных городских условиях

Заключительный этап работы состоит в апробации задачи последнего километра в искусственных городских условиях. Процедура апробации может быть проведена без изменений, повторяя принцип в симуляции, описанный в главе 7.1. В качестве искусственных городских условий подразумевается два окружения, разных по уровню сложности и характеру.

Первое искусственное окружение планируется организовать внутри учебного помещения КФУ (Спортзал УНИКСа, полигон в Инженерном институте). В качестве объектов, отвечающих городской среде, могут быть использованы копии дорожных знаков, предоставленных компанией-производителем (см. Рисунок 8.3.1), а также другие дорожные элементы (экспериментальный малоразмерный полигон), предоставленные компанией-производителем. Дополнительно необходимо реализовать искусственную городскую топологию, используя панели для имитации жилых и офисных зданий.

Вторым искусственным окружением может быть использована площадь около учебных зданий КФУ. В этом случае также необходимо использовать инвентарь, который бы имитировал городскую среду. Идея использования такого окружения является его приближенность к реальным условиям, так как эксперименты будут происходить в реальной среде с динамическими и естественными препятствиями (дерево, бордюр, ограждение).

Рисунок 8.3.1 - Дорожные знаки для задач распознавания

ЗАКЛЮЧЕНИЕ

Данная работа предлагает концепцию решения проблемы последнего километра. Проблема последнего километра описывает задачу доставки товара из единой транспортной точки в конечную точку назначения. Так как серьезные затраты на осуществление доставок в конечную точку являются проблемой многих транспортных компаний, в настоящий момент разрабатываются пути решения этой проблемы. Одно из инновационных направлений – использование роботов в качестве автономных интеллектуальных агентов, которые будут доставлять товары в несколько точек в населенном пункте.

Робот «Юниор» является малоразмерным мобильным роботом с рулевой системой управления. Такая система управления присутствует в полноразмерных автомобилях. Дополнительно, в робомобиле присутствует верхняя крышка, под которой располагается пространство для возможного хранения товаров для доставки. Такой мобильный робот является потенциальным решением для поставленной задачи. Однако для её решения необходимо разработать соответствующее базовое и специфичное цели программное обеспечение робота: создание модели робота, настройка системы навигации, апробация решения последнего километра в симуляции. Необходимость разработки базового ПО заключается в том, что компания-производитель предоставляет минимальный набор ROS-пакетов, которые позволяют только получать данные с датчиков и посылать команды на моторы робота.

Для создания модели робота использовался симулятор Gazebo, интегрированный в ROS. Созданная модель соответствует реальным размерам робота, его массовым характеристикам элементами, наличию сенсоров. По завершении этапа моделирования была подготовлена и написана научная публикация для международной конференции «International Conference on

Artificial Life and Robotics» ICAROB [58]. Статья была принята к печати и будет индексирована в БД Scopus.

Далее для модели робота был настроен Система навигации (глобальный планировщик, локальный планировщик, конфигурации карт стоимостей), проведены первичные эксперименты в симуляции. Для будущей апробации задачи последнего километра был создан виртуальный мир с имитацией городской среды и девятью различными целями в нем.

Предложенное решение демонстрирует концепт автоматизированного последнего километра и может быть развит как полноценный проект на реальном роботе «Юниор». Идеи и план по дальнейшей реализации рассмотрены в главе 8. Весь исходный код, использованная литература и статья для ICAROB 2019 загружены в систему контроля версий GitLab: http://gititis.kpfu.ru/XEN/lirs-development-junior-car.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Siciliano B., Oussama K. Handbook of robotics. Springer, 2016.
2. BMW says self-driving car to be Level 5 capable by 2021 [Электронный ресурс] // Automotive News: [сайт]. [2017]. URL: https://www.autonews.com/article/20170316/MOBILITY/170319877/bmw-says-self-driving-car-to-be-level-5-capable-by-2021 (дата обращения: 10.05.2019).
3. Future of Driving [Электронный ресурс] // Tesla: [сайт]. [2019]. URL: https://www.tesla.com/autopilot?redirect=no (дата обращения: 01.05.2019).
4. Беспилотник на дорогах Сколково [Электронный ресурс] // Яндекс Такси: [сайт]. [2018]. URL: https://taxi.yandex.ru/blog/bespilotnik-v-skolkovo (дата обращения: 02.05.2019).
5. Waymo [Электронный ресурс] // Википедия - свободная энциклопедия: [сайт]. URL: https://en.wikipedia.org/wiki/Waymo (дата обращения: 03.05.2019).
6. Apollo Open Platform [Электронный ресурс] // Apollo: [сайт]. [2018]. URL: http://apollo.auto/ (дата обращения: 04.05.2019).
7. Uber [Электронный ресурс] // Uber: [сайт]. URL: https://www.uber.com/info/atg/technology/ (дата обращения: 05.05.2019).
8. Christensen H., Okamura A., Mataric M., Kumar V., Hager G., Choset H. Next generation robotics // arXiv preprint arXiv:1606.09205, 2016.
9. Wen J., He L., Zhu F. Swarm Robotics Control and Communications: Imminent Challenges for Next Generation Smart Logistics // IEEE Communications Magazine, Vol. 56, No. 7, 2018. pp. 102-107.

10 Brunner G., Szebedy B., Tanner S., Wattenhofer R. The Urban Last Mile Problem: Autonomous Drone Delivery to Your Balcony // arXiv preprint arXiv:1809.08022, 2018.

11 Marks M. Robots in Space: Sharing Our World with Autonomous Delivery Vehicles // Available at SSRN 3347466, 2019.

12 Jain D., Sharma Y. Adoption of next generation robotics: A case study on Amazon // A Case Research Journal, Vol. 3, 2017. pp. 9-23.

13 Starship [Электронный ресурс] // Starship: [сайт]. URL: https://www.starship.xyz/ (дата обращения: 15.05.2019).

14 Marble [Электронный ресурс] // Marble: [сайт]. [2019]. URL: https://www.marble.io/ (дата обращения: 15.05.2019).

15 Bite the futue [Электронный ресурс] // Kiwibot: [сайт]. [2019]. URL: https://www.kiwicampus.com/ (дата обращения: 16.05.2019).

16 Brunner G., Szebedy B., Tanner S., Wattenhofer R. The Urban Last Mile Problem: Autonomous Drone Delivery to Your Balcony // arXiv preprint arXiv:1809.08022, 2018.

17 Chong Z., Qin B., Bandyopadhyay T., Wongpiromsarn T., Rankin E., Ang M., Frazzoli E., Rus D., Hsu D., Low K. Autonomous personal vehicle for the first- and last-mile transportation services // IEEE 5th International Conference on Cybernetics and Intelligent Systems (CIS). 2011. pp. 253-260.

18 Murai R., Sakai T., Kawano H., Matsukawa Y., Kitano Y., Honda Y., Campbell K. A novel visible light communication system for enhanced control of autonomous delivery robots in a hospital // IEEE/SICE International Symposium on System Integration (SII). 2012. pp. 510-516.

19. Wu J., Chaoshun L., Lijun Z., Ruifeng L., Guanglin W. Design and implementation of an omnidirectional mobile robot platform with unified I/O interfaces // In Mechatronics and Automation (ICMA), 2017 IEEE International Conference on. 2017. pp. 410-415.

20. Feng Y., Chengjun D., Xia L., Xinghua Z. Integrating Mecanum wheeled omni-directional mobile robots in ROS // In Robotics and Biomimetics (ROBIO), 2016 IEEE International Conference on. 2016. pp. 643-648.

21. Wu P., Kai W., Juzhong Z., Qi Z. Optimal Design for PID Controller Based on DE Algorithm in Omnidirectional Mobile Robot // MATEC Web of Conferences. 2017. Vol. 95. P. 08014.

22. Yip H.M., Ho K.K., Chu M.H.A., Lai K.W. Development of an omnidirectional mobile robot using a RGB-D sensor for indoor navigation // The 4th Annual IEEE International Conference on Cyber Technology in Automation, Control and Intelligent. 2014. pp. 162-167.

23. Doroftei I., Grosu V., Spinu V. Omnidirectional mobile robot-design and implementation // Bioinspiration and Robotics Walking and Climbing Robots. 2007.

24. Kanjanawanishkul K. Omnidirectional wheeled mobile robots: wheel types and practical applications // International Journal of Advanced Mechatronic Systems, Vol. 6, No. 6, 2015. pp. 289-302.

25. Ismael O.Y., Hedley J. Analysis, design, and implementation of an omnidirectional mobile robot platform // American Scientific Research Journal for Engineering, Technology, and Sciences (ASRJETS), Vol. 22, No. 1, 2016. pp. 195-209.

26. Ren C., Ma S. Analysis and control of an omnidirectional mobile robot // IEEE ISR 2013. 2013. pp. 1-6.

27 Ren C., Ma S. Dynamic modeling and analysis of an omnidirectional mobile robot // 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems. 2013. pp. 4860-4865.

28 Diegel O., Aparna B., Glen B., Johan P., Sylveste T. Improved mecanum wheel design for omni-directional robots // In Proc. 2002 Australasian Conference on Robotics and Automation. 2002. pp. 117-121.

29 Ramirez-Serrano A., Roman K. Modified mecanum wheels for traversing rough terrains // 2010 Sixth International Conference on Autonomic and Autonomous Systems. 2010. pp. 97-103.

30 Salih J.E.M., Mohamed R., Sazali Y., Abdul H.A., Mohd R.M. Designing omni-directional mobile robot with mecanum wheel // American Journal of Applied Sciences, Vol. 3, No. 5, 2006. pp. 1831-1835.

31 Tapia J., Eric W., Patrick B., Aldo J., Ethan C., John P., Dan C., Mo J., Benjamin C. Autonomous mobile robot platform with multi-variant task-specific end-effector and voice activation // 2016 World Automation Congress (WAC). 2016. pp. 1-6.

32 Sarmento L., Francisco N., Ricardo S.M., João S., João, Sena Esteve. Remote control system for a mobile platform with four Mecanum wheels // International Journal of Mechatronics and Applied Mechanics, No. 1, 2017. pp. 274-281.

33 Lam H.K. Doing good across organizational boundaries: Sustainable supply chain practices and firms' financial risk // International Journal of Operations & Production Management, Vol. 38, No. 12, 2018. pp. 2389-2412.

34 Tjahjono B., Esplugues C., Ares E., Pelaez G. What does industry 4.0 mean to supply chain? // Procedia Manufacturing, Vol. 13, 2017. pp. 1175-1182.

35 Singh N. Emerging technologies to support supply chain management // Communications of the ACM, Vol. 46, No. 9, 2003. pp. 243-247.

36 Alicke K., Rexhausen D., Seyfert A. Supply chain 4.0 in consumer goods // Mckinsey & Company, 2017.

37 Проект "Каргобот" [Электронный ресурс] // Avrora Robotics: [сайт]. [2018]. URL: https://avrora-robotics.com/ru/projects/passenger-logistics/ (дата обращения: 27.04.2019).

38 Quigley M., Conley K., Gerkey B., Faust J., Foote T., Leibs J., Wheeler R., Ng A.Y. ROS: an open-source Robot Operating System // ICRA workshop on open source software. Kobe, Japan. 2009. Vol. 3. P. 5.

39 Sagitov A., Gerasimov Y. Towards DJI Phantom 4 Realistic Simulation with Gimbal and RC Controller in ROS/Gazebo Environment // 10th International Conference on Developments in eSystems Engineering (DeSE). 2017. pp. 262-266.

40 Afanasyev I., Sagitov A., Magid E. ROS-based SLAM for a Gazebo-simulated mobile robot in image-based 3D model of indoor environment // International Conference on Advanced Concepts for Intelligent Vision Systems. 2015. pp. 273-283.

41 Sokolov M., Afanasyev I., Lavrenov R., Sagitov A., Sabirova L., Magid E. Modelling a crawler-type UGV for urban search and rescue in Gazebo environment // Artificial Life and Robotics (ICAROB 2017), International Conference on. 2017. pp. 360-362.

42 Pepper C., Balakirsky S., Scrapper C. Robot simulation physics validation // Proceedings of the 2007 Workshop on Performance Metrics for Intelligent Systems. 2007. pp. 97-104.

43. Pecka M., Zimmermann K., Svoboda T. Fast simulation of vehicles with non-deformable tracks // 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). 2017. pp. 6414-6419.

44. Yao W., Dai W., Xiao J., Lu H., Zheng Z. A simulation system based on ros and gazebo for robocup middle size league // IEEE international conference on robotics and biomimetics (ROBIO). 2015. pp. 54-59.

45. Qian W., Xia Z., Xiong J., Gan Y., Guo Y., Weng S., Deng H., Hu Y., Zhang J. Manipulation task simulation using ROS and Gazebo // IEEE International Conference on Robotics and Biomimetics (ROBIO 2014). 2014. pp. 2594-2598.

46. Kohlbrecher S., Meyer J., Graber T., Petersen K., Von Stryk O., Klingauf U. Robocuprescue 2014-robot league team hector darmstadt (Germany) // RoboCupRescue 2014, 2014.

47. Kermorgant O. A dynamic simulator for underwater vehicle-manipulators // International Conference on Simulation, Modeling, and Programming for Autonomous Robots. 2014. pp. 25-36.

48. Rohmer E., Singh S.P., Freese M. V-REP: A versatile and scalable robot simulation framework // 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems. 2013. pp. 1321-1326.

49. Ziegler J.G., Nichols N.B. Optimum settings for automatic controllers // trans. ASME, Vol. 64, No. 11, 1942.

50. Planning for car-like robots [Электронный ресурс] // Ros.org: [сайт]. URL: http://wiki.ros.org/teb_local_planner/Tutorials/Planning%20for%20car-like%20robots (дата обращения: 20.01.2019).

51. Rosmann C., Hoffmann F., Bertram T. Kinodynamic trajectory optimization and control for car-like robots // 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). 2017. pp. 5681-5686.

52. Rosmann C., Feiten W., Wosch T., Hoffmann F., Bertram T. Trajectory modification considering dynamic constraints of autonomous robots // ROBOTIK 2012; 7th German Conference on Robotics. 2012. pp. 1-6.

53. Rosmann C., Feiten W., Wosch T., Hoffmann F., Bertram T. Efficient trajectory optimization using a sparse model // 2013 European Conference on Mobile Robots. 2013. pp. 138-143.

54. Quinlan S., Khatib O. Elastic bands: Connecting path planning and control // Proceedings IEEE International Conference on Robotics and Automation. 1993. pp. 802-807.

55. Rosmann C., Hoffmann F., Bertram T. Integrated online trajectory planning and optimization in distinctive topologies // Robotics and Autonomous Systems, Vol. 88, 2017. pp. 142-153.

56. Rosmann C., Hoffmann F., Bertram T. Planning of multiple robot trajectories in distinctive topologies // 2015 European Conference on Mobile Robots (ECMR). 2015. pp. 1-6.

57. Limpert N., Schiffer S., Ferrein A. A local planner for Ackermann-driven vehicles in ROS SBPL // 2015 Pattern Recognition Association of South Africa and Robotics and Mechatronics International Conference (PRASA-RobMech). 2015. pp. 172-177.

58. Shabalina K., Sagitov A., Su K., Hsia K.H., Magid E. Avrora Unior Car-like Robot in Gazebo Environment // International Conference on Artificial Life and Robotics (ICAROB). 2019 (in press). pp. 116-119.

59 Move_base [Электронный ресурс] // Ros.org: [сайт]. URL: http://wiki.ros.org/move_base?distro=melodic (дата обращения: 25.01.2019).

60 Рулевое управление [Электронный ресурс] // MirFord.ru: [сайт]. [2017]. URL: http://www.mirford.ru/ford-mondeo/166.htm (дата обращения: 15.02.2019).

61 ПИД-регулятор [Электронный ресурс] // Википелия - Свободная энциклопедия: [сайт]. URL: https://ru.wikipedia.org/wiki/%D0%9F%D0%98%D0%94-%D1%80%D0%B5%D0%B3%D1%83%D0%BB%D1%8F%D1%82%D0%BE%D1%80 (дата обращения: 20.12.2018).

ПРИЛОЖЕНИЕ А

Описание модели робота «Юниор»

А.1 Фрагмент файла junior_car.urdf.xacro

```
<?xml version="1.0"?>
<robot name="junior_car"
  xmlns:xacro="http://www.ros.org/wiki/xacro">
  <xacro:include filename="$(find junior_car)/urdf/junior_car.gazebo.xacro" />
  <!-- inertial macros -->
  <xacro:include filename="$(find junior_car)/urdf/inertia.xacro" />
  <!-- add transmissions -->
  <xacro:include filename="$(find junior_car)/urdf/junior_car.transmission.xacro" />
  <link name="footprint">
   </link>
  <link name="base_link">
    <visual>
      <origin xyz="0 -0.02 0.165" rpy="0 0 0" />
      <geometry>
        <mesh filename="package://junior_car/meshes/base_link.STL" />
      </geometry>
      <material name="">
       <!--<color rgba="0.25098 0.25098 0.25098 1" />-->
      </material>
    </visual>
    <collision>
      <origin xyz="0 -0.02 0.165" rpy="0 0 0" />
      <geometry>
        <mesh filename="package://junior_car/meshes/base_link.STL"/>
      </geometry>
    </collision>
    <xacro:box_inertial_origin x="${b_height}" y="${b_width}" z="${b_depth}" mass="${b_mass}"
origin="0 -0.025 0.2" rpy="0 0 1.5708"/>
  </link>
```

```
  <joint name="footprint_joint" type="fixed">
    <parent link="footprint"/>
    <child link="base_link"/>
    <origin xyz="0 0 0" rpy="0 0 -1.5709" />
   </joint>
  <link name="hokuyo_sensor">
    <xacro:box_inertial x="${hokuyo_depth}" y="${hokuyo_depth}" z="${hokuyo_width}"
mass="${hokuyo_mass}"/>
    <visual>
     <origin xyz="0 0 0" rpy="0 -${pi/2} ${pi}" />
     <geometry>
       <mesh filename="package://junior_car/meshes/hokuyo_sensor.STL" />
     </geometry>
     <material name="">
       <color rgba="0.250980392156863 0.250980392156863 0.250980392156863 1" />
     </material>
    </visual>
    <collision>
     <origin xyz="0 0 0" rpy="0 -${pi/2} ${pi}" />
     <geometry>
       <mesh filename="package://junior_car/meshes/hokuyo_sensor.STL" />
     </geometry>
    </collision>
  </link>
```

A.2 Фрагмент файла junior_car.transmission.xacro

```
<?xml version="1.0"?>
<robot xmlns:xacro="http://www.ros.org/wiki/xacro">
<xacro:macro name="junior_transmission">
      <transmission name="junior_car_tran_1">
          <robotNamespace>/junior_car</robotNamespace>
          <type>transmission_interface/SimpleTransmission</type>
          <joint name="right_steer_joint">
             <hardwareInterface>PositionJointInterface</hardwareInterface>
```

```
        </joint>
        <actuator name="junior_car_motor_1">
          <hardwareInterface>PositionJointInterface</hardwareInterface>
          <mechanicalReduction>1</mechanicalReduction>
        </actuator>
      </transmission>
```

A.3 Фрагмент файла junior_car.gazebo.xacro

```
<?xml version="1.0"?>
<robot xmlns:xacro="http://www.ros.org/wiki/xacro">
 <gazebo>
  <plugin name="gazebo_ros_control" filename="libgazebo_ros_control.so">
   <robotNamespace>/</robotNamespace>
  <robotSimType>steer_bot_hardware_gazebo/SteerBotHardwareGazebo</robotSimType>
  </plugin>
 </gazebo>
 <gazebo reference="imu_link">
  <material>Gazebo/Yellow</material>
  <gravity>true</gravity>
  <sensor name="imu_sensor" type="imu">
   <always_on>true</always_on>
   <update_rate>100</update_rate>
   <visualize>true</visualize>
   <topic>/avrora_imu</topic>
   <plugin filename="libgazebo_ros_imu_sensor.so" name="imu_plugin">
    <topicName>imu</topicName>
    <bodyName>imu_link</bodyName>
    <updateRateHZ>10.0</updateRateHZ>
    <gaussianNoise>0.0</gaussianNoise>
    <xyzOffset>0 0 0</xyzOffset>
    <rpyOffset>0 0 0</rpyOffset>
    <frameName>imu_link</frameName>
   </plugin>
   <pose>0 0 0 0 0 0</pose>
```

```
    </sensor>
  </gazebo>
```

A.4 Фрагмент файла inertia.xacro

```
<?xml version="1.0"?>
<robot xmlns:xacro="http://www.ros.org/wiki/xacro">
  <property name="M_PI" value="3.1415926535897931" />
  <xacro:property name="hokuyo_mass" value="0.16"/>
  <xacro:property name="hokuyo_height" value="0.05"/>
  <xacro:property name="hokuyo_depth" value="0.05"/>
  <xacro:property name="hokuyo_width" value="0.07"/>
  <xacro:macro name="sphere_inertial" params="radius mass">
    <inertial>
      <mass value="${mass}" />
      <origin xyz="0 0 0" />
      <inertia ixx="${0.4 * mass * radius * radius}" ixy="0.0" ixz="0.0" iyy="${0.4 * mass * radius *
radius}" iyz="0.0" izz="${0.4 * mass * radius * radius}" />
    </inertial>
  </xacro:macro>
  <xacro:macro name="cylinder_inertial" params="radius length mass origin rpy">
    <inertial>
      <mass value="${mass}" />
      <origin xyz="${origin}" rpy="${rpy}" />
      <inertia ixx="${0.0833333 * mass * (3 * radius * radius + length * length)}" ixy="0.0" ixz="0.0"
iyy="${0.0833333 * mass * (3 * radius * radius + length * length)}" iyz="0.0" izz="${0.5 * mass * radius *
radius}" />
    </inertial>
  </xacro:macro>
```

ПРИЛОЖЕНИЕ Б

Программирование контроля модели робота «Юниор»

```
#include "ros/ros.h"
#include "std_msgs/Float64.h"
#include <utility>
#include <cmath>
#include <sstream>
#include <string>
#include "ackermann_msgs/AckermannDriveStamped.h"
ros::Publisher left_pub;
ros::Publisher right_pub;
std_msgs::Float64 fi_r;
std_msgs::Float64 fi_l;
double const lateral_distance = 0.7;
double const front_distance = 0.54;
double const pi2 = 1.5708;
double radiusCalc (double angle)
{   return lateral_distance/tan(angle); }
double formulaCalc (double radius, bool sign)
{  if (sign)
  {  return lateral_distance / (radius + (front_distance/2)); }
  else
  {  return lateral_distance / (radius - (front_distance/2)); }  }
void steeringCallback(const ackermann_msgs::AckermannDriveStamped::ConstPtr &msg)
{  double radius = radiusCalc(msg->drive.steering_angle);
  ROS_INFO("---Calculated radius : %.4f,", radius);
  if (msg->drive.steering_angle >= 0 && msg->drive.steering_angle <= 1.20)
  {  fi_r.data = atan(formulaCalc(radius, false));
    fi_l.data = atan(formulaCalc(radius,true));
    ROS_INFO("---Calculated angles : left wheel  %.4f, right wheel %.4f", fi_r.data,fi_l.data);
    left_pub.publish(fi_r);
```

```
    right_pub.publish(fi_l); }
  if (msg->drive.steering_angle <= 0 && msg->drive.steering_angle >= -1.20)
  {   ROS_INFO("---Calculated angles : right wheel %.4f, left wheel %.4f", fi_r.data,fi_l.data);
      fi_r.data = atan(formulaCalc(radius,true));
      fi_l.data = atan(formulaCalc(radius, false));
      left_pub.publish(fi_l);
      right_pub.publish(fi_r);  }
  if (msg->drive.steering_angle > 1.20)
  { double radius_exception(1.2);
    ROS_INFO("---Calculated radius exception : %.4f,", radius);
    fi_r.data = atan(formulaCalc(radius_exception, false));
    fi_l.data = atan(formulaCalc(radius_exception,true));
    ROS_INFO("---Calculated angles : left wheel  %.4f, right wheel %.4f", fi_r.data,fi_l.data);
    left_pub.publish(fi_r);
    right_pub.publish(fi_l); }
  if (msg->drive.steering_angle < -1.2)
  { double radius_exception(-1.2);
    ROS_INFO("---Calculated radius exception : %.4f,", radius);
    fi_r.data = atan(formulaCalc(radius_exception, false));
    fi_l.data = atan(formulaCalc(radius_exception,true));
    ROS_INFO("---Calculated angles : left wheel  %.4f, right wheel %.4f", fi_r.data,fi_l.data);
    left_pub.publish(fi_r);
    right_pub.publish(fi_l); }
}
int main(int argc, char **argv)
{ ros::init(argc, argv, "steer_remap");
  ros::NodeHandle n;
  std::string lw_topic, rw_topic, central_steer_topic;
  central_steer_topic = "/junior_car/ackermann_cmd";
  lw_topic = "/junior_car/SteerLeft_controller/command";
  rw_topic = "/junior_car/SteerRight_controller/command";
  if (n.getParam("left_wheel_steer_topic", lw_topic))
  {  ROS_INFO("Got param: %s", lw_topic.c_str());}
```

```
  if (n.getParam("right_wheel_steer_topic", rw_topic))
  { ROS_INFO("Got param: %s", rw_topic.c_str()); }
  if (n.getParam("right_wheel_steer_topic", central_steer_topic))
  { ROS_INFO("Got param: %s", central_steer_topic.c_str()); }
  left_pub = std::move(n.advertise<std_msgs::Float64>(lw_topic, 1024));
  right_pub = std::move(n.advertise<std_msgs::Float64>(rw_topic, 1024));
  ros::Subscriber sub = n.subscribe(central_steer_topic, 1024, steeringCallback);
  ros::spin();
  return 0;
}
```

ПРИЛОЖЕНИЕ В

Нода корректировки данных с лазерного дальномера модели робота «Юниор»

```
#include "ros/ros.h"
#include "std_msgs/Float64.h"
#include <utility>
#include <cmath>
#include <sstream>
#include <string>
#include "sensor_msgs/LaserScan.h"
ros::Subscriber sub;
ros::Publisher pub;
const double laser_incline = 0.261799;
const double laser_height = 0.4;
const double max_obstacle = laser_height/tan(laser_incline);
void scanCallback(const sensor_msgs::LaserScan::ConstPtr& scan)
{
  sensor_msgs::LaserScan msg;
  msg.header.stamp = scan->header.stamp;
  msg.header.frame_id = "fake_laser";
  msg.angle_min = scan->angle_min;
  msg.angle_max = scan->angle_max;
  msg.angle_increment = scan->angle_increment;
  msg.time_increment = scan->time_increment;
  msg.range_min = scan->range_min;
  msg.range_max = scan->range_max;
  msg.ranges.resize(scan->ranges.size());
  msg.intensities.resize(scan->intensities.size());
  for (int i=0; i<scan->ranges.size() ; i++)
  {
    double r = scan->ranges[i] * cos(laser_incline);
    if (r > max_obstacle)
```

```
      r = std::numeric_limits<double>::infinity();
    msg.ranges[i] = r;
    msg.intensities[i] = scan->intensities[i];
  }
  pub.publish(msg);
}
int main(int argc, char **argv)
{
  ros::init(argc, argv, "rectificated_laser");
  ros::NodeHandle n;
  std::string scan_topic, remap_topic;
  remap_topic = "/rectificated_scan";
  scan_topic="/scan";
  pub = n.advertise<sensor_msgs::LaserScan>(remap_topic, 50);
  sub = n.subscribe(scan_topic, 1, scanCallback);
  ros::spin();
  return 0;
}
```

ПРИЛОЖЕНИЕ Г

Система навигации модели робота «Юниор»

Г.1 Содержание файла junior_hector.launch

```
<launch>
  <node pkg="move_base" type="move_base" respawn="false" name="move_base" output="screen">
    <rosparam file="$(find junior_2dnav)/config/dagny_nav.yaml" command="load" />
    <rosparam file="$(find junior_2dnav)/config/teb_local_planner.yaml" command="load"/>
    <param name="SBPLLatticePlanner/primitive_filename" value="$(find
junior_2dnav)/primitives/dagny.mprim" />
    <rosparam file="$(find junior_2dnav)/config/common_costmap.yaml" command="load"
ns="local_costmap"/>
    <rosparam file="$(find junior_2dnav)/config/local_costmap.yaml" command="load"
ns="local_costmap"/>
    <rosparam file="$(find junior_2dnav)/config/common_costmap.yaml" command="load"
ns="global_costmap"/>
    <rosparam file="$(find junior_2dnav)/config/global_costmap.yaml" command="load"
ns="global_costmap"/>
    <remap from="cmd_vel" to="/steer_drive_controller/cmd_vel" />
    <remap from="odom" to="/steer_drive_controller/odom" />
  </node>
</launch>
```

Г.2 Фрагмент файла hector.launch

```
<launch>
 <arg name="tf_map_scanmatch_transform_frame_name" default="scanmatcher_frame"/>
 <arg name="base_frame" default="footprint"/>
 <arg name="odom_frame" default="odom"/>
 <arg name="pub_map_odom_transform" default="false"/>
 <arg name="scan_subscriber_queue_size" default="5"/>
 <arg name="scan_topic" default="rectificated_scan"/>
 <arg name="map_size" default="2048"/>
 <node pkg="hector_mapping" type="hector_mapping" name="hector_mapping" output="screen">
  <!-- Frame names -->
```

```
    <param name="map_frame" value="map" />
    <param name="base_frame" value="$(arg base_frame)" />
    <param name="odom_frame" value="$(arg odom_frame)" />
    <!-- Tf use -->
    <param name="use_tf_scan_transformation" value="true"/>
    <param name="use_tf_pose_start_estimate" value="false"/>
    <param name="pub_map_odom_transform" value="$(arg pub_map_odom_transform)"/>
```

Г.3 Содержание файла junior_nav.launch

```
base_global_planner: SBPLLatticePlanner
recovery_behaviors:
  [
    {
      name: conservative_reset,
      type: clear_costmap_recovery/ClearCostmapRecovery,
    },
  ]
planner_patience: 15.0
clearing_rotation_allowed: false
base_local_planner: teb_local_planner/TebLocalPlannerROS
controller_frequency: 10.0
SBPLLatticePlanner:
  allocated_time: 15
  initial_epsilon: 10.0
```

Г.4 Фрагмент файла teb_local_planner.yaml

```
TebLocalPlannerROS:
 odom_topic: odom
 map_frame: /map
 # Trajectory
 teb_autosize: True
 dt_ref: 0.6
 dt_hysteresis: 0.1
 global_plan_overwrite_orientation: True
```

```
 max_global_plan_lookahead_dist: 3.0
 feasibility_check_no_poses: 0
 # Robot
max_vel_x: 53
 max_vel_x_backwards: -53
 max_vel_theta: 53
 acc_lim_x: 53
 acc_lim_theta: 53
 acc_lim_y: 53
 min_turning_radius: 1.0
 footprint_model:
   type: "polygon"
   vertices: [[-0.556, -0.325],[-0.556, 0.325],[0.556, 0.325],[0.556, -0.325]]
 # GoalTolerance
 xy_goal_tolerance: 0.2
 yaw_goal_tolerance: 0.3
 free_goal_vel: False
 # Obstacles
 min_obstacle_dist: 0.7
 include_costmap_obstacles: True
 costmap_obstacles_behind_robot_dist: 1.5
 obstacle_poses_affected: 30
 inflation_dist: 0.6
 costmap_converter_plugin: ""
 costmap_converter_spin_thread: True
 costmap_converter_rate: 7 #was 5
```

Г.5 Содержание файла common_costmap.yaml

```
footprint: [[-0.556, -0.325],[-0.556, 0.325],[0.556, 0.325],[0.556, -0.325]]
footprint_padding: 0.0
transform_tolerance: 0.1
robot_base_frame: footprint
global_frame: map
update_frequency: 5.0
```

```
publish_frequency: 1.0
rolling_window: true
static:
  map_topic: map
  subscribe_to_updates: true
plugins:
  - { name: obstacles_laser, type: "costmap_2d::ObstacleLayer" }
  - { name: inflation, type: "costmap_2d::InflationLayer" }
resolution: 0.05
obstacles_laser:
  observation_sources: laser
  laser:
    {
      data_type: LaserScan,
      clearing: true,
      marking: true,
      topic: rectificated_scan,
      inf_is_valid: true,
    }
inflation_layer:
  inflation_radius: 0.7
  enabled: true
footprint_layer:
  enabled: true
sonar_layer:
  enabled: true
```

Г.6 Содержание файла local_costmap.yaml

```
static_map: false
width: 6.0
height: 6.0
resolution: 0.05
```

Г.7 Содержание файла global_costmap.yaml

```
track_unknown_space: true
```

```
resolution: 0.10

update_frequency: 5.0

publish_frequency: 5.0
```

Г.8 Содержание файла rviz_config.launch

```
<?xml version="1.0"?>

<launch>

  <arg name="rviz"    default="true" />

  <arg name="rviz_file" default="$(find junior_2dnav)/rviz/junior_rviz.rviz" />

  <node pkg="rviz" type="rviz" name="$(anon rviz)" args="-d $(arg rviz_file)" output="screen" if="$(arg
rviz)"/>

</launch>
```

Г.9 Фрагмент файла junior_rviz.rviz

```
Panels:

  - Class: rviz/Displays

    Help Height: 78

    Name: Displays

    Property Tree Widget:

      Expanded:

        - /Global Options1

        - /Status1

        - /TF1/Frames1

        - /Odometry1/Shape1

      Splitter Ratio: 0.5

    Tree Height: 775

  - Class: rviz/Selection

    Name: Selection

  - Class: rviz/Tool Properties

    Expanded:

      - /2D Pose Estimate1

      - /2D Nav Goal1

      - /Publish Point1

    Name: Tool Properties

    Splitter Ratio: 0.588679016
```

- Class: rviz/Views